

中国科学院研究生教学丛书

第二版

高级人工智能

史忠植

科学出版社

内容简介

人工智能是计算机科学的一个分支，是一门研究机器智能的学科，即用人工的方法和技术，研制智能机器或智能系统来模仿、延伸和扩展人的智能，实现智能行为。

本书第二版共 16 章，首先讨论人工智能的认知问题和逻辑基础，然后论述约束推理、定性推理、基于范例推理、概率推理。第七章至第十三章重点讨论机器学习，包括归纳学习、支持向量机、解释学习、强化学习、粗糙集、关联规则、知识发现。第十四章阐述分布式智能。最后两章分别讨论进化计算和人工生命。与第一版相比，增加了五章新内容。其他章节也作了较大的修改和补充。

本书内容新颖，反映该领域的最新研究进展，特别总结了作者多年的科研成果。全书力求从理论、算法、系统、应用讨论人工智能的方法和关键技术。本书可以作为信息领域和相关专业的高等院校高年级学生和研究生教材，也可以供有关科技人员学习参考。

图书在版编目 (CIP) 数据

高级人工智能/史忠植 —北京：科学出版社， 2006.8

责任编辑

ISBN 7-03-005984-0

北京东黄城根北街 16 号

邮政编码：100717

科学出版社出版

北京东黄城根北街 16 号

邮政编码：100717

中国科学院印刷厂印刷

新华书店北京发行所发行 各地新华书店经销

*

2006 年 8 月第 二 版 开本：787 1092 1/16

2006 年 8 月第一次印刷 印张：21 1/2

印数：1—6000 册 字数：582 千字

定价：38.00 元

前 言

人工智能的长期目标是建立人类水平的人工智能。人工智能诞生 50 年来，在崎岖不平的道路上取得了可喜的进展，特别与机器学习、数据挖掘、计算机视觉、专家系统、自然语言处理、规划和机器人等相关的应用带来了良好的经济效益和社会效益。广泛使用的互联网也正在探索应用知识表示和推理，构建语义 Web，提高互联网信息的效用率。信息化的必然趋势是智能化，智能革命将开创人类后文明史。如果说蒸汽机创造了工业社会，那么智能机也一定能奇迹般地创造出智能社会，实现社会生产的自动化和智能化，促进知识密集型经济的大发展。

人工智能是计算机科学的一个分支，是一门研究机器智能的学科，即用人工的方法和技术，研制智能机器或智能系统来模仿、延伸和扩展人的智能，实现智能行为。人工智能一般可分为符号智能和计算智能。符号智能是传统人工智能，它以物理符号系统为基础，研究知识表示、获取、推理过程。运用知识解决问题是当前符号智能的最基本、最重要的特点，因此人们经常把当前的人工智能称作为知识工程阶段。知识工程侧重研究知识信息处理的方法和技术，促进人工智能的发展。

计算智能包括神经计算、模糊系统、遗传算法、进化规划等。要实现智能革命，就要更深入地了解人的大脑。彻底揭开人脑的奥秘，是自然科学面临的最大挑战之一。21 世纪初，美国国家科学基金会（NSF）和美国商务部（DOC）共同资助了一个雄心勃勃的计划——“提高人类素质的聚合技术”（Convergent Technology for Improving Human Performance），将纳米技术、生物技术、信息技术和认知科学看作 21 世纪四大前沿技术，并将认知科学视为最优先发展领域，主张这四大技术融合发展，并描绘了这样的科学前景：“聚合技术以认知科学为先导，因为一旦我们能够在如何（how）、为何（why）、何处（where）、何时（when）这四个层次上理解思维，我们就可以用纳米科技来制造它，用生物技术和生物医学来实现它，最后用信息技术来操纵和控制它，使它工作”。这将对人类社会带来巨大影响。在欧盟的第 6 次研究与技术发展框架最新一轮的研究经费资助中，在大脑科学研究方面投入了 4500 万欧元的资金。在 2007 年开始执行的第 7 次发展框架中，欧洲议会还会进一步增加对大脑研究的投入。

表面看来，符号智能和神经计算是完全不同的研究方法，前者基于知识，后者基于数据；前者采用推理，后者通过映射。1996 年，Minsky 在第四届太平洋地区人工智能国际会议的特邀报告“Computers, Emotions and Common Sense”中认为，神经计算与符号计算可以结合起来，神经网络是符号系统的基础。致力于混合系统研究的人们，正是从这种结合出发，提出各种可能的设想。这与我们提出的人类思维的层次模型是吻合的。

本书自 1998 年发行以来，受到了读者广泛的欢迎，许多院校用作教科书或教材，众多研究人员用作参考书。在这近十年的时间里，人工智能又取得了许多重要的进展，特别在不确定推理、机器学习、多主体系统、人工生命等方面，广大读者迫切要求出版第二版。

智能主体(intelligent agent)通过与环境交互，实现智能行为。智能主体从环境接受感

知信息，进行协同工作，执行各种智能行为。20 世纪 90 年代以来，多主体系统成为人工智能研究的核心之一。

人工生命是指用计算机和精密机械等生成或构造表现自然生命系统行为特点的仿真系统或模型系统。人工生命是形成新的信息处理体系强大的推动力，并成为研究生物的一个特别有用的工具。人工生命的研究可能将信息科学和生命科学结合起来，形成生命信息科学，成为人工智能研究新的途径。

本书第二版在第一版基础上作了重大修订，共分 16 章。第一章是绪论，从人工智能的认知问题出发，介绍本书撰写的指导思想，概要介绍人工智能当前研究的热点。第二章讨论人工智能逻辑，较系统地讨论非单调逻辑和与智能主体有关的逻辑系统。第三章讨论约束推理，介绍许多实用的约束推理方法。第四章介绍定性推理，着重讨论几种重要的定性推理方法。多年来作者及其领导的集体一直从事基于范例推理的研究，其主要成果构成第五章。概率推理是一种重要的不确定推理，第六章给予重点讨论。机器学习是当前人工智能研究的核心，也是知识发现、数据挖掘等领域的重要基础，本书用 7 章的篇幅给予论述，反映研究的最新进展。第七章论述归纳学习。第八章介绍支持向量机。第九章讨论解释学习。第十章论述强化学习。第十一章介绍粗糙集理论。第十二章阐述关联规则。集成的知识发现系统给出在第十三章。近几年来分布智能取得重要进展，结合我们研究的成果，在第十四章重点讨论主体理论和多主体系统的关键技术。在第十五章，对进化计算进行了探讨，重点讨论遗传算法。最后一章探讨人工生命，概要介绍人工生命研究的内容和取得的进展。与上一版相比，增加了许多新内容，包括概率推理、支持向量机、强化学习、粗糙集、关联规则等。其他章节也在上一版的基础上作了修改，补充了许多新内容，以反映最新研究成果。本书第二版在每章都增加了习题，以便读者加深理解课文内容。

本书是根据中国科学院研究生院（北京）计算机科学技术专业博士、硕士研究生教学的需要，作者于 1994 年开设了高级人工智能课程。教材经不断修改和扩充，于 1998 年正式出版，定为普通高等教育“九五”国家级重点教材、中国科学院研究生教学丛书。本书实际上是集体研究成果的总结，先后有 6 位博士后、40 多位博士生、90 多位硕士生参加了研究。陆汝钤院士、戴汝为院士、李衍达院士、张钹院士、董韫美院士、高庆狮院士、林惠敏院士、陈霖院士、郭爱克院士、李国杰院士、何新贵院士、郑南宁院士、钟义信教授、石纯一教授、涂序彦教授、张成奇教授、王珏研究员、何华灿教授等给予诚挚的帮助和有益的讨论，作者愿借此机会深表谢意。

本书有关研究得到国家自然科学基金、国家重大基础研究计划、国家 863 高技术计划、北京市自然科学基金、中国科学院知识创新工程等的资助。科学出版社对本书的出版给予了大力支持，在此一并致谢。

史忠植

2006 年 3 月

目 录

第一章 绪 论.....	1
1.1 人工智能的渊源.....	1
1.2 人工智能的认知问题.....	3
1.3 思维的层次模型.....	4
1.4 符号智能.....	5
1.5 人工智能的研究方法.....	7
1.5.1 认知学派.....	7
1.5.2 逻辑学派.....	7
1.5.3 行为学派.....	8
1.6 自动推理.....	8
1.7 机器学习.....	10
1.8 分布式人工智能.....	12
1.9 人工思维模型.....	14
1.10 知识系统.....	15
习题.....	17
第二章 人工智能逻辑.....	19
2.1 概述.....	19
2.2 逻辑程序设计.....	21
2.2.1 逻辑程序定义.....	21
2.2.2 Prolog 数据结构和递归.....	22
2.2.3 SLD 归结.....	23
2.2.4 非逻辑成分: CUT.....	25
2.3 非单调逻辑.....	27
2.4 封闭世界假设.....	29
2.5 默认逻辑.....	30
2.6 限制逻辑.....	35
2.7 非单调逻辑 NML.....	38
2.8 自认知逻辑.....	40
2.8.1 Moore 系统 \mathcal{L}_B	40
2.8.2 $0\mathcal{L}$ 逻辑.....	40
2.8.3 标准型定理.....	42
2.8.4 \Diamond -记号以及稳定扩张的一种判定过程.....	43
2.9 真值维护系统.....	45
2.10 情景演算的逻辑基础.....	50
2.10.1 刻划情景演算的多类逻辑.....	51
2.10.2 LR 中的基本动作理论.....	52

2.11 框架问题.....	53
2.11.1 框架公理.....	53
2.11.2 框架问题解决方案的准则.....	54
2.11.3 框架问题的非单调解决方案.....	56
2.12 动态描述逻辑 DDL.....	60
2.12.1 描述逻辑.....	60
2.12.2 动态描述逻辑的语法.....	62
2.12.3 动态描述逻辑的语义.....	64
习题.....	67

第三章 约束推理..... 69

3.1 概 述.....	69
3.2 回溯法.....	73
3.3 约束传播.....	74
3.4 约束传播在树搜索中的作用.....	76
3.5 智能回溯与真值维护.....	76
3.6 变量例示次序与赋值次序.....	77
3.7 局部修正搜索法.....	77
3.8 基于图的回跳法.....	78
3.9 基于影响的回跳法.....	79
3.10 约束关系运算的处理.....	82
3.10.1 恒等关系的单元共享策略.....	82
3.10.2 区间传播.....	84
3.10.3 不等式图.....	84
3.10.4 不等式推理.....	85
3.11 约束推理系统 COPS.....	86
3.12 ILOG SOLVER.....	89
习题.....	94

第四章 定性推理..... 95

4.1 概 述.....	95
4.2 定性推理的基本方法.....	95
4.3 定性模型推理.....	96
4.4 定性进程推理.....	98
4.5 定性仿真推理.....	101
4.5.1 定性状态转换.....	102
4.5.2 QSIM 算法.....	102
4.6 代数方法.....	104
4.7 几何空间定性推理.....	105
4.7.1 空间逻辑.....	105
4.7.2 空间时间关系描述.....	107
4.7.3 空间和时间逻辑的应用.....	108

4.7.4 Randell 算法	109
习题.....	110

第五章 基于范例的推理..... 111

5.1 概述.....	111
5.2 类比的形式定义.....	112
5.3 过程模型.....	113
5.4 范例的表示.....	115
5.4.1 语义记忆单元.....	115
5.4.2 记忆网.....	116
5.5 范例的索引.....	118
5.6 范例的检索.....	118
5.7 相似性关系.....	120
5.7.1 语义相似性.....	120
5.7.2 结构相似性.....	120
5.7.3 目标特征.....	121
5.7.4 个体相似性.....	121
5.7.5 相似性计算.....	122
5.8 范例的复用.....	123
5.9 范例的保存.....	124
5.10 基于例示的学习.....	125
5.10.1 基于例示学习的任务.....	125
5.10.2 IB1 算法	126
5.10.3 降低存储要求.....	128
5.11 范例工程	130
5.12 范例约简算法	132
5.13 中心渔场预报专家系统	134
5.13.1 问题分析与范例表示.....	135
5.13.2 相似性度量.....	136
5.13.3 索引与检索.....	137
5.13.4 基于框架的修正.....	138
5.13.5 实验结果.....	139
习题.....	140

第六章 概率推理..... 141

6.1 概述.....	141
6.1.1 贝叶斯网络的发展历史.....	141
6.1.2 贝叶斯方法的基本观点.....	141
6.1.3 贝叶斯网络在数据挖掘中的应用.....	142
6.2 贝叶斯概率基础	144
6.2.1 概率论基础.....	144
6.2.2 贝叶斯概率.....	146

6.3 贝叶斯问题求解	148
6.3.1 几种常用的先验分布选取方法	149
6.3.2 计算学习机制	151
6.3.3 贝叶斯问题求解步骤	153
6.4 简单贝叶斯学习模型	155
6.4.1 简单贝叶斯学习模型	155
6.4.2 简单贝叶斯模型的提升	157
6.4.2 提升简单贝叶斯分类的计算复杂性	160
6.5 贝叶斯网络的建造	160
6.5.1 贝叶斯网络的结构及建立方法	160
6.5.2 学习贝叶斯网络的概率分布	161
6.5.3 学习贝叶斯网络的网络结构	163
6.6 贝叶斯潜在语义模型	166
6.7 半监督文本挖掘算法	170
6.7.1 网页聚类	170
6.7.2 对含有潜在类别主题词的文档的类别标注	171
6.7.3 基于简单贝叶斯模型学习标注和未标注样本	172
习题	176

第七章 归纳学习..... 177

7.1 概 述	177
7.2 归纳学习的逻辑基础	178
7.2.1 归纳学习的一般模式	178
7.2.2 概念获取的条件	179
7.2.3 问题背景知识	180
7.2.4 选择型和构造型泛化规则	182
7.3 偏置变换	184
7.4 变型空间方法	185
7.4.1 消除候选元素算法	186
7.4.2 两种改进算法	188
7.5 AQ 归纳学习算法	189
7.6 CLS 学习算法	190
7.6 ID3 学习算法	191
7.6.1 信息论简介	191
7.6.2 信息论在决策树学习中得以应用	191
7.6.3 ID3 算法	192
7.6.4 ID3 算法应用举例	193
7.6.5 连续性属性离散化	195
7.7 基于偏置变换的决策树学习算法 BSDT	196
7.7.1 偏置的形式化	196
7.7.2 表示偏置变换	197
7.7.3 算法描述	198
7.7.4 过程偏置变换	199

7.7.5 基于偏置变换的决策树学习算法 BSDT	202
7.7.6 经典范例库维护算法 TCBM	202
7.7.7 偏置特征抽取算法	203
7.7.8 改进的决策树生成算法 GSD	204
7.7.9 实验结果	205
7.8 归纳学习的计算理论	207
7.8.1 Gold 学习理论	207
7.8.2 模型推理系统	208
7.8.3 Valiant 学习理论	209
习题	210

第八章 支持向量机..... 212

8.1 统计学习问题	212
8.1.1 经验风险	212
8.1.2 VC 维	212
8.2 学习过程的一致性	213
8.2.1 学习一致性的经典定义	213
8.2.2 学习理论的重要定理	213
8.2.3 VC 熵	214
8.3 结构风险最小归纳原理	215
8.4 支持向量机	218
8.4.1 线性可分	218
8.4.2 线性不可分	220
8.5 核函数	222
8.5.1 多项式核函数	222
8.5.2 径向基函数	222
8.5.3 多层感知机	223
8.5.4 动态核函数	223
8.6 基于分类超曲面的海量数据分类方法	225
8.6.2 SVM 直接方法基本思想	226
8.6.3 实现算法	227
8.6.4 实验结果分析	228
习题	231

第九章 解释学习..... 233

9.1 概述	233
9.2 解释学习模型	234
9.3 解释泛化学习方法	235
9.3.1 基本原理	235
9.3.2 解释与泛化交替进行	237
9.4 全局取代解释泛化方法	238
9.5 解释特化学习方法	241

9.6 解释泛化的逻辑程序	243
9.6.1 工作原理	243
9.6.2 元解释器	244
9.6.3 实验例子	245
9.7 基于知识块的 SOAR 系统	246
9.8 可操作性标准	249
9.8.1 PRODIGY 的效用问题	250
9.8.2 SOAR 系统的可操作性	251
9.8.3 MRS-EBG 的可操作性	251
9.8.4 META-LEX 的处理方法	252
9.9 不完全领域知识下的解释学习	252
9.9.1 不完全领域知识	252
9.9.2 逆归结方法	252
9.9.3 基于深层知识方法	254
习题	255

第十章 强化学习..... 256

10.1 概 述	256
10.2 强化学习模型	257
10.3 动态规划	260
10.4 蒙特卡罗方法	260
10.5 时序差分学习	262
10.6 Q 学习	264
10.7 强化学习中的函数估计	266
10.8 强化学习的应用	267
习题	269

第十一章 粗糙集..... 271

11.1 概 述	271
11.1.1 知识的分类观点	272
11.1.2 新型的隶属关系	273
11.1.3 概念的边界观点	274
11.2 知识的约简	274
11.2.1 一般约简	274
11.2.2 相对约简	275
11.2.3 知识的依赖性	276
11.3 决策逻辑	276
11.3.1 决策表的公式化定义	276
11.3.2 决策逻辑语言	277
11.3.3 决策逻辑语言的语义	278
11.3.4 决策逻辑的推演	279
11.3.5 规范表达形式	280

11.3.6 决策规则和决策算法	280
11.3.7 决策规则中的一致性和不分明性	281
11.4 决策表的约简	281
11.4.1 属性的依赖性	281
11.4.2 一致决策表的约简	281
11.4.3 非一致决策表的约简	286
11.5 粗糙集的扩展模型	290
11.6 粗糙集的实验系统	292
11.7 粒度计算	294
11.7 粗糙集的展望	294
习题	295

第十二章 关联规则..... 296

12.1 关联规则挖掘概述	296
12.1.1 关联规则的意义和度量	296
12.1.2 经典的挖掘算法	298
12.2 广义模糊关联规则的挖掘	300
12.3 挖掘关联规则的数组方法	303
12.4 任意多表间关联规则的并行挖掘	304
12.4.1 问题的形式描述	304
12.4.2 单表内大项集的并行计算	305
12.4.3 任意多表间大项集的生成	306
12.4.4 跨表间关联规则的提取	307
12.5 基于分布式系统的关联规则挖掘算法	308
12.5.1 候选集的生成	309
12.5.2 候选数据集的本地剪枝	310
12.5.3 候选数据集的全局剪枝	312
12.5.4 合计数轮流检测	314
12.5.5 分布式挖掘关联规则的算法	314
12.6 词性标注规则的挖掘算法与应用	317
12.6.1 汉语词性标注	317
12.6.2 问题的描述	318
12.6.3 挖掘算法	319
12.6.4 试验结果	321
习题	322

第十三章 知识发现..... 324

13.1 概述	324
13.2 知识发现的任务	326
13.2.1 数据总结	326
13.2.2 概念描述	327
13.2.3 分类	327

13.2.4 聚类	327
13.2.5 相关性分析	328
13.2.6 偏差分析	328
13.2.7 建模	328
13.3 知识发现工具	329
13.4 MSMiner 的体系结构	331
13.4.1 数据挖掘模型	331
13.4.2 系统功能	333
13.4.3 体系结构	333
13.5 元数据管理	334
13.4.1 MSMiner 元数据的内容	334
13.4.2 MSMiner 元数据库	335
13.4.3 MSMiner 元数据对象模型	336
13.6 数据仓库	339
13.6.1 数据仓库含义	339
13.6.2 MSMiner 数据仓库的基本结构	340
13.6.3 主题	342
13.6.4 数据抽取和集成	342
13.6.5 数据抽取和集成的元数据	345
13.6.6 数据仓库建模及 OLAP 的实现	347
13.7 算法库管理	351
13.7.1 数据挖掘算法的元数据	3511
13.7.2 可扩展性的实现	352
13.7.3 采掘算法的接口规范	353
习题	355

第十四章 分布智能..... 356

14.1 概述	356
14.2 分布式问题求解	357
14.2.1 分布式问题求解系统分类	358
14.2.2 分布式问题求解过程	358
14.3 主体基础	360
14.4 主体理论	361
14.4.1 理性主体	361
14.4.2 BDI 主体模型	361
14.4 主体结构	362
14.4.1 主体基本结构	362
14.4.2 慎思主体	363
14.4.3 反应主体	366
14.4.4 混合结构	368
14.4.5 INTERRAP	369
14.4.6 MAPE 主体结构	370
14.5 主体通信语言 ACL	380

14.5.1 主体间通信概述	381
14.5.2 FIPA ACL 消息	382
14.5.3 交互协议	386
14.5.4 ACL 语义学的形式化基础	387
14.6 协调和协作	389
14.6.1 引言	389
14.6.2 合同网	391
14.6.3 部分全局规划	393
14.6.4 基于约束传播的规划	395
14.6.5 基于生态学的协作	403
14.6.6 基于对策论的协商	404
14.6.7 基于意图的协商	405
14.7 移动主体	405
14.8 多主体环境 MAGE	407
14.8.1 MAGE 系统框架结构	407
14.8.2 主体统一建模语言 AUML	408
14.8.3 可视化主体开发环境 VASstudio	408
14.8.4 MAGE 运行平台	409
习题	410

第十五章 进化计算..... 412

15.1 概述	412
15.2 进化系统理论的形式模型	413
15.3 达尔文进化算法	415
15.4 分类器系统	416
15.5 桶链算法	420
15.6 遗传算法	421
15.6.1 遗传算法的主要步骤	422
15.6.2 表示模式	423
15.6.3 杂交操作	424
15.6.4 变异操作	426
15.6.5 反转操作	426
15.7 并行遗传算法	426
15.8 分类器系统 BOOLE	427
15.9 规则发现系统	430
15.10 进化策略	433
15.11 进化规划	433
习题	433

第十六章 人工生命..... 435

16.1 引言	435
16.2 人工生命的探索	438

16.3 人工生命模型	439
16.4 人工生命的研究方法	441
16.5 细胞自动机	444
16.6 形态形成理论	446
16.7 混沌理论	447
16.8 人工生命的实验系统	448
16.8.1 Tierra 数字生命进化模型	448
16.8.2 Avida	449
16.8.3 Terrarium 生物饲养生态系统	450
16.8.4 人工鱼	451
16.8.5 AutoLife	451
习题	452
参考文献	454

第1章 绪论

1.1 人工智能的渊源

人工智能(Artificial Intelligence)主要研究用人工的方法和技术,模仿、延伸和扩展人的智能,实现机器智能。2005 年, McCarthy 指出人工智能的长期目标是实现人类水平的人工智能 [McCarthy 2005]

产业革命解放了人的体力劳动,使用机器可以完成繁重的体力工作,极大地促进了人类社会的进步和经济的发展。在人类历史发展的过程中,自然会提出如何用机器放大人的脑力劳动。制造和使用仿人的智能机器是人们长期以来的愿望。

我国曾经发明了不少智能工具和机器。例如,算盘是应用广泛的古典计算机;水运仪象台是天文观测与星象分析仪器;候风地动仪是测报与显示地震的仪器。我们祖先提出的阴阳学说蕴涵着丰富的哲理,对现代逻辑的发展有重大影响。

在国外, Aristotle (公元前 384—322) 在《工具论》的著作中提出形式逻辑。Bacon (1561—1626) 在《新工具》中提出归纳法。Leibnitz (1646—1716) 研制了四则计算器,提出了“通用符号”和“推理计算”的概念,使形式逻辑符号化,可以说是“机器思维”研究的萌芽。

19 世纪以来,数理逻辑、自动机理论、控制论、信息论、仿生学、计算机、心理学等科学技术的进展,为人工智能的诞生,准备了思想、理论和物质基础。Boole (1815—1864) 创立了布尔代数,他在《思维法则》一书中,首次用符号语言描述了思维活动的基本推理法则。Godel (1906—1978) 提出了不完备性定理。Turing (1912—1954) 提出了理想计算机模型——图灵机,创立了自动机理论。1943 年, McCulloch 和 Pitts 提出了 MP 神经网络模型,开创了人工神经网络的研究。1946 年, Manochly 和 Eckert 研制成功 ENIAC 电子数字计算机。1948 年, Wiener 创立了控制论, Shannon 创立了信息论。

现实世界中,相当多的问题求解是复杂的,常常没有算法可遵循,或者即使有计算方法,也是 NP 问题。人们可以采用启发式知识进行求解,把复杂的问题大大简化,可在浩瀚的搜索空间中迅速找到解答。这是运用专门领域的经验知识,经常会取得有关问题的满意解,但不是数学上的最优解。这种处理问题的方法具有显著的特色,导致人工智能的诞生。1956 年由 McCarthy、Minsky 等发起,美国的几位心理学家、数学家、计算机科学家、信息论学家在 Dartmouth 大学举办夏季讨论会,正式提出人工智能的术语,开始了具有真正意义的人工智能的研究。经过三十多年的研究和发展,人工智能取得了很大的进展。许多人工智能专家系统研制成功,并投入使用。自然语言理解、机器翻译、模式识别、机器人、图像处理等方面,取得了不少研究成果。其应用渗透到许多领域,促进它们的发展。

在 20 世纪 50 年代,人工智能以博弈、游戏为对象进行研究。1956 年, Samuel 研制成功

具有自学习能力的启发式博弈程序。同年, Newell、Simon 等研制了启发式程序 Logic Theorist, 证明了《数学原理》书中 38 条定理, 开创了利用计算机研究思维活动规律的工作。Chomsky 提出了语言文法, 开创了形式语言的研究。1958 年 McCarthy 建立了人工智能程序设计语言 LISP, 不仅可以处理数值, 而且可以更方便地处理符号, 为人工智能研究提供了重要工具。

20 世纪 60 年代前期, 人工智能以搜索算法、通用问题求解 (GPS) 的研究为主。1963 年, Newell 发表了问题求解程序, 使启发式程序有更大的普适性。1961 年, Minsky 发表题为“走向人工智能的步骤”的论文, 推动了人工智能的发展。1965 年 Feigenbaum 研制成功了 DENDRAL 化学专家系统, 使人工智能的研究从着重算法转向知识表示的研究, 也是人工智能研究走向实用化的标志。1965 年, Robinson 提出了归结原理。1968 年 Quillian 提出了语义网络的知识表示方法。1969 年, 国际人工智能联合会 (IJCAI) 成立。从 1969 年起, 每两年召开一次国际人工智能学术会议, 由 IJCAI 主办的人工智能学报 “Artificial Intelligence” 于 1970 年创刊。

20 世纪 70 年代前期, 人工智能研究以自然语言理解、知识表示为主。1972 年, Winograd 发表了自然语言理解系统 SHRDLU。法国马赛大学的 Colmerauer 创建了 PROLOG 语言。1973 年, Schank 提出了概念从属理论。1974 年, Minsky 提出了重要的框架知识表示法。1977 年, Feigenbaum 在第五届国际人工智能会议上提出了知识工程。他认为, 知识工程是人工智能的原理和方法, 对那些需要专家知识才能解决的应用难题提供求解的手段。恰当运用专家知识的获取、表示和推理过程的构成与解释, 是设计基于知识系统的重要技术问题。

20 世纪 80 年代, 人工智能蓬勃发展。专家系统开始广泛应用, 出现了专家系统开发工具, 开始兴起人工智能产业。特别是 1982 年, 日本政府正式宣布投资开发第五代计算机, 极大地推动了人工智能的发展。许多国家制订相应的计划, 进行人工智能和智能计算机系统的研究。我国也将智能计算机系统的研究列入国家 863 高技术计划。

五十年来, 人工智能的研究取得了一定的进展, 提出了启发式搜索策略、非单调推理、机器学习的方法等。人工智能应用, 特别是专家系统、智能决策、智能机器人、自然语言理解等方面的成就促进了人工智能的研究。当前, 以知识信息处理为中心的知识工程是人工智能的显著标志。

人工智能的研究与其它事物发展一样, 出现过曲折。从一开始, 人工智能的工作者因过分地乐观而受人指责。20 世纪 60 年代初, 人工智能的创始人 Simon 等就乐观地预言:

- (1) 十年内数字计算机将是世界象棋冠军。
- (2) 十年内计算机将证明一个未发现的重要的数学定理。
- (3) 十年内数字计算机将谱写具有相当美学价值的而为批评家所认可的乐曲。
- (4) 十年内大多数心理学理论将采用计算机程序的形式。

这些预言至今还未完全实现。甚至连一个 3 岁的小孩也能轻而易举地从一幅图画中辨别出一棵树来, 而功能最强大的超级计算机也只能在小孩认树方面达到中等水平。对小孩故事的理解还非常困难。

人工智能尚缺乏必要的理论。在一些关键技术方面，诸如机器学习、非单调推理、常识性知识表示、不确定推理等尚未取得突破性的进展。人工智能对全局性判断模糊信息处理、多粒度视觉信息的处理是极为困难的。

总的来看，人工智能还处于智能学科研究的早期阶段，必须开展智能科学的研究。智能科学研究智能的基本理论和实现技术，是由脑科学、认知科学、人工智能等学科构成的交叉学科。脑科学从分子水平、细胞水平、行为水平研究人脑智能机理，建立脑模型，揭示人脑的本质。认知科学是研究人类感知、学习、记忆、思维、意识等人脑心智活动过程的科学。人工智能研究用人工的方法和技术，模仿、延伸和扩展人的智能，实现机器智能。三门学科共同研究，探索智能科学的新概念、新理论、新方法，必将在 21 世纪共创辉煌[史忠植 2006]。

1.2 人工智能的认知问题

一般认为，认知(cognition)是和情感、动机、意志等相对的理智或认识过程。美国心理学家 Houston 等人将对“认知”的看法归纳为如下五种主要类型：

- (1) 认知是信息的处理过程；
- (2) 认知是心理上的符号运算；
- (3) 认知是问题求解；
- (4) 认知是思维；
- (5) 认知是一组相关的活动，如知觉、记忆、思维、判断、推理、问题求解、学习、想象、概念形成、语言使用等。

认知心理学家 Dodd 等则认为，认知应包括三个方面，即适应、结构和过程。也就是说，认知是为了一定的目的，在一定的心理结构中进行的信息加工过程。

认知科学是研究人类感知和思维信息处理过程的科学，包括从感觉的输入到复杂问题求解，从人类个体到人类社会的智能活动，以及人类智能和机器智能的性质[史忠植 1990]。认知科学是现代心理学、信息科学、神经科学、数学、科学语言学、人类学乃至自然哲学等学科交叉发展的结果。它是人工智能重要的理论基础。

认知科学的兴起和发展标志着对以人类为中心的认知和智能活动的研究已进入到新的阶段。认知科学的研究将使人类自我了解和自我控制，把人的知识和智能提高到空前未有的高度。它的研究将为智能革命、知识革命和信息革命建立理论基础，为智能计算机系统的研制提供新概念、新思想、新途径。

受到纽威尔和西蒙早期研究工作的推动，认知科学的研究在 50 年代末期就出现了[司马贺 1986]。认知科学家的研究成果提供了较好的模型以代替行为主义学说关于人的简化模型。认知科学研究的目的是要说明和解释人在完成认知活动时是如何进行信息加工的。认知科学涉及的问题非常广泛，包括知觉、语言、学习、记忆、思维、问题求解、创造、注意，以及环境、社会文化背景对认知的影响。

1991 年,有代表性的杂志“Artificial Intelligence”第 47 卷发表了人工智能基础专辑,指出了人工智能研究的趋势。Kirsh 在专辑中提出了人工智能的五个基本问题[Kirsh 1991]:

- (1) 知识与概念化是否是人工智能的核心?
- (2) 认知能力能否与载体分开来研究?
- (3) 认知的轨迹是否可用类自然语言来描述?
- (4) 学习能力能否与认知分开来研究?
- (5) 所有的认知是否有一种统一的结构?

这些问题都是与人工智能有关的认知问题,必须从认知科学的基础理论进行探讨。这些问题都涉及人工智能的关键,因此成为不同学派的分水岭。各个学派对上述问题都有不同的答案。

1.3 思维的层次模型

思维是客观现实的反映过程,是具有意识的人脑对于客观现实的本质属性、内部规律性的自觉的、间接的和概括的反映。由于科学的发展和对思维研究的结果,当代已进入一个注重自知的阶段,强调自我认识。1984 年,钱学森教授倡导开展思维科学(Noetic Science)的研究[钱学森 1986]。

人类思维的形态主要有感知思维、形象思维、抽象思维和灵感思维。感知思维是一种初级的思维形态。在人们开始认识世界时,只是把感性材料组织起来,使之构成有条理的知识,所能认识到的仅是现象。在此基础上形成的思维形态即是感知思维。人们在实践过程中,通过眼、耳、鼻、舌、身等感官直接接触客观外界而获得的各种事物的表面现象的初步认识,它的来源和内容都是客观的、丰富的。

形象思维主要是用典型化的方法进行概括,并用形象材料来思维,是一切高等生物所共有的。形象思维是与神经机制的连接论相适应的。模式识别、图像处理、视觉信息加工都属于这个范畴。

抽象思维是一种基于抽象概念的思维形式,通过符号信息处理进行思维。只有语言的出现,抽象思维才成为可能,语言和思维互相促进,互相推动。可以认为物理符号系统是抽象思维的基础。

对灵感思维至今研究甚少。有人认为,灵感思维是形象思维扩大到潜意识,人脑有一部分对信息进行加工,但是人并没有意识到。也有人认为,灵感思维是顿悟。灵感思维在创造性思维中起重要作用,有待进行深入研究。

人的思维过程中,注意发挥重要作用。注意使思维活动有一定的方向和集中,保证人能够及时地反映客观事物及其变化,使人能够更好地适应周围环境。注意限制了可以同时思考的数目。因此在有意识的活动中,大脑更多地表现为串行的。而看和听是并行的。

根据上述讨论,作者提出人类思维的层次模型(参见图 1.1)[史忠植 1990;Shi 1992; Shi 1994]。图中感知思维是极简单的思维形态,它通过人的眼、耳、鼻、舌、身感知器官产生表象,形成初级的思维。形象思维以神经网络的连接论为理论基础,可以高度并行处理。抽象思维以物理符号系统为理论基础,用语言表述抽象的概念。由于注意的作用,使其处理基本

上是串行的。

思维模型就是要研究这三种思维形式的相互关系，以及它们之间的相互转换的微观过程。人们可以用神经网络的稳定吸引子来表示联想记忆、图像识别的问题。但是要解决从形象思维到逻辑思维的过渡的微过程，还需要作长期的进一步研究。

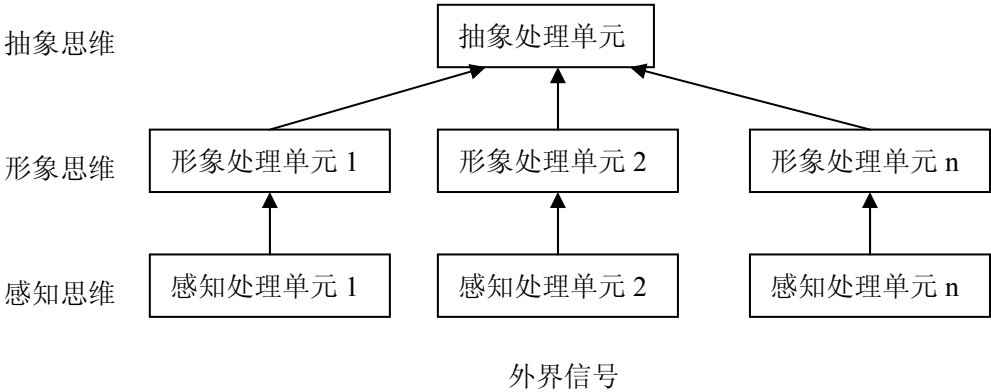


图 1.1 思维的层次模型

1.4 符号智能

智能是什么？智能是个体有目的的行为、合理的思维，以及有效的适应环境的综合性能。通俗地说，智能是个体认识客观事物和运用知识解决问题的能力。人类个体的智能是一种综合性能，具体讲，可以包括感知与认识客观事物、客观世界与自我的能力；通过学习取得经验、积累知识的能力；理解知识、运用知识和运用经验分析问题和解决问题的能力；联想、推理、判断、决策的能力；运用语言进行抽象、概括的能力；发现、发明、创造、创新的能力；实时地、迅速地、合理地应付复杂环境的能力；预测、洞察事物发展变化的能力等。人生活在社会中，其智能与社会环境有密切的关系。随着人类社会的不断进步，智能的概念也不断发展。

人工智能(Artificial Intelligence)是相对人的自然智能而言，即用人工的方法和技术，模仿、延伸和扩展人的智能，实现某些“机器思维”。作为一门学科，人工智能研究智能行为的计算模型，研制具有感知、推理、学习、联想、决策等思维活动的计算系统，解决需要人类专家才能处理的复杂问题。

长期来，人们从人脑思维的不同层次对人工智能进行研究，形成了符号主义、连接主义和行为主义。传统人工智能是符号主义，它以Newell和Simon提出的物理符号系统假设为基础。物理符号系统假设认为物理符号系统是智能行为充分和必要的条件。物理符号系统由一组符号实体组成，它们都是物理模式，可在符号结构的实体中作为组分出现。该系统可以进行建立、修改、复制、删除等操作，以生成其它符号结构。

连接主义研究非程序的、适应性的、大脑风格的信息处理的本质和能力。人们也称它为

神经计算。由于它近年来的迅速发展,大量的神经网络的机理、模型、 算法不断地涌现出来。神经网络主体是一种开放式的神经网络环境,提供典型的、具有实用价值的神经网络模型。系统采用开放方式,使得新的网络模型可以比较方便地进入系统中,利用系统提供良好的用户界面和各种工具,对网络算法进行调试修改。另外,对已有的网络模型的改善也较为简单,为新的算法的实现提供了良好的环境。

神经计算从脑的神经系统结构出发来研究脑的功能,研究大量简单的神经元的集团信息处理能力及其动态行为。其研究重点侧重于模拟和实现人的认识过程中的感知觉过程、形象思维、分布式记忆和自学习自组织过程。特别是对并行搜索、联想记忆,时空数据统计描述的自组织以及一些相互关联的活动中自动获取知识,更显示出了其独特的能力,并普遍认为神经网络适合于低层次的模式处理。

神经网络的基本特点集中表现在:①以分布式方式存储信息。②以并行方式处理信息。③具有自组织、自学习能力[史忠植 1993b]。正是这些特点,使神经网络为人们在利用机器加工处理信息方面提供了一种全新的方法和途径。当然,随着人工神经网络应用的深入,人们也发现原有的模型和算法所存在的问题,在理论的深入也碰到很多原来非线性理论、逼近论中的难点。可是我们相信,在深入、广泛应用的基础上,这个领域将会继续发展,并会对科学技术有很大的促进作用。我们提出的神经场理论是一种新的尝试。

目前,符号处理系统和神经网络模型的结合是一个重要的研究方向。模糊神经网络就是将模糊逻辑、神经网络等结合在一起,在理论、方法和应用上发挥各自的优势,设计出具有一定学习能力、动态获取知识能力的系统。

Brooks 提出了无需知识表示的智能[Brooks 1991a],无需推理的智能[Brooks 1991b]。他认为智能只是在与环境的交互作用中表现出来,在许多方面是行为心理学观点在现代人工智能中的反映,人们称为基于行为的人工智能,简言之,称为行为主义。

这三种研究从不同侧面研究人的自然智能,与人脑思维模型有其对应关系。粗略地划分,可以认为符号主义研究抽象思维,连接主义研究形象思维,而行为主义研究感知思维。表 1.1 给出了符号主义、连接主义和行为主义特点的比较。

表 1.1 符号主义、连接主义和行为主义特点的比较

	符号主义	连接主义	行为主义
认识层次	离散	连续	连续
表示层次	符号	连接	行动
求解层次	自顶向下	由底向上	由底向上
处理层次	串行	并行	并行
操作层次	推理	映射	交互
体系层次	局部	分布	分布
基础层次	逻辑	模拟	直觉判断

有人把人工智能形成两大类:一类是符号智能,一类是计算智能。符号智能是以知识为基础,通过推理进行问题求解。也即所谓的传统人工智能。计算智能是以数据为基础,通过训练建立联系,进行问题求解。人工神经网络、遗传算法、模糊系统、进化程序设计、人工生命等都可以包括在计算智能。

目前，传统人工智能主要运用知识进行问题求解。从实用观点看，人工智能是一门知识工程学：以知识为对象，研究知识的表示方法、知识的运用和知识获取。本书主要介绍和讨论传统人工智能的内容。有关计算智能的内容请参阅史忠植的《神经计算》等书。

1.5 人工智能的研究方法

从 20 世纪 50 年代以来，人工智能经过发展，形成了许多学派。不同学派的研究方法、学术观点、研究重点有所不同。这里，仅以认知学派、逻辑学派、行为学派为重点，介绍人工智能的研究方法。

1.5.1 认知学派

以 Simon, Minsky 和 Newell 等为代表，从人的思维活动出发，利用计算机进行宏观功能模拟。20 世纪 50 年代，Newell 和 Simon 等共同倡导“启发式程式”。他们编制了称为“Logic Theorist”的计算机程序，模拟人证明数学定理的思维过程。60 年代初，他们又研制了“通用问题求解程序(General Problem Solver”，简称 GPS，分三个阶段模拟了人在解题过程中的一般思维规律：首先拟订初步解题计划；然后利用公理、定理和规则，按规划实施解题过程；不断进行“目的 — 手段”分析，修订解题规划；从而使 GPS 具有一定的通用性。

1976 年 Newell 和 Simon 提出了物理符号系统假设，认为物理系统表现智能行为必要和充分的条件是它是一个物理符号系统。这样，可以把任何信息加工系统看成是一个具体的物理系统，如人的神经系统、计算机的构造系统等。所谓符号就是模式。任何一个模式，只要它能和其它模式相区别，它就是一个符号。不同的英文字母就是不同的符号。对符号进行操作就是对符号进行比较，即找出哪几个是相同的符号，哪几个是不同的符号。物理符号系统的基本任务和功能是辨认相同的符号和区分不同的符号。

20 世纪 80 年代 Newell 等又致力于 SOAR 系统的研究。SOAR 系统是以知识块(Chunking)理论为基础，利用基于规则的记忆，获取搜索控制知识和操作符，实现通用问题求解。

Minsky 从心理学的研究出发，认为人们在他们日常的认识活动中，使用了大批从以前的经验中获取并经过整理的知识。该知识是以一种类似框架的结构记存在人脑中。因此，在 20 世纪 70 年代他提出了框架知识表示方法。到 20 世纪 80 年代，Minsky 认为人的智能，根本不存在统一的理论。1985 年，他发表了一本著名的书《Society of Mind(思维社会)》。书中指出思维社会是由大量具有某种思维能力的单元组成的复杂社会。

1.5.2 逻辑学派

逻辑学派是以 McCarthy 和 Nilsson 等为代表，主张用逻辑来研究人工智能，即用形式化的方法描述客观世界。他们认为：

- (1) 智能机器必须有关于自身环境的知识。
- (2) 通用智能机器要能陈述性地表达关于自身环境的大部分知识。
- (3) 通用智能机器表示陈述性知识的语言至少要有一阶逻辑的表达能力。

逻辑学派在人工智能研究中，强调的是概念化知识表示、模型论语义、演绎推理等。McCarthy 主张任何事物都可以用统一的逻辑框架来表示，在常识推理中以非单调逻辑为中心。

1.5.3 行为学派

人工智能的研究大部分是建立在一些经过抽象的、过份简单的现实世界模型之上，Brooks 认为应走出这种抽象模型的象牙塔，而以复杂的现实世界为背景，让人工智能理论、技术先经受解决实际问题的考验，并在这种考验中成长。

Brooks 提出了无需知识表示的智能 (Brooks 1991a)，无需推理的智能 (Brooks 1991b)。他认为智能只是在与环境的交互作用中表现出来，其基本观点：

- (1) 到现场去；
- (2) 物理实现；
- (3) 初级智能；
- (4) 行为产生智能。

以这些观点为基础，Brooks 研制了一种机器虫，用一些相对独立的功能单元，分别实现避让、前进、平衡等功能，组成分层异步分布式网络，取得了一定程度的成功，特别是对机器人的研究开创了一种新的方法。

不同的人人工智能学派，对基本的认知问题给以不同的回答。以 Nilsson 为代表的逻辑学派，对认知问题的 (1) — (4) 给予肯定的回答，对 (5) 持中立观点。以 Newell 为代表的认知学派，对认知问题的 (1)、(3)、(5) 给予肯定的回答。而以 Brooks 为代表的行为学派，对认知问题 (1) — (5) 均持否定的看法。

1.6 自动推理

从一个或几个已知的判断(前提)逻辑地推论出一个新的判断(结论)的思维形式称为推理，这是事物的客观联系在意识中的反映。人解决问题就是利用以往的知识，通过推理得出结论。自动推理的理论和程序是程序推导、程序正确性证明、专家系统、智能机器人等研究领域的重要基础。

自动推理早期的工作主要集中在机器定理证明。开创性的工作是 Simon 和 Newell 的 Logic Theorist。1956 年 Robinson 提出归结原理，把自动推理的研究向前推进了一步。归结法推理规则简单，而且在逻辑上是完备的，因而成为逻辑式程序设计语言 Prolog 的计算模型。后来又出现了自然演绎法和等式重写式等。这些方法在某些方面优于归结法，但它们本

质上都存在组合问题，都受到难解性的制约。

从任何一个实用系统来说，总存在着很多非演绎的部分，因而导致了各种各样推理算法的兴起，并削弱了企图为人工智能寻找一个统一的基本原理的观念。从实际的观点来看，每一种推理算法都遵循其特殊的、与领域相关的策略，并倾向于使用不同的知识表示技术。从另一方面来说，如果能找到一个统一的推理理论，当然是很有用的。人工智能理论研究的一个很强的推动力就是要设法寻找更为一般的、统一的推理算法。

人工智能自动推理研究的成果之一是非单调逻辑的发明。这是一种伪演绎系统。所谓非单调推理，指的是一个正确的公理加到理论中，反而会使预先所得的一些结论变得无效了。非单调推理明显地比单调推理复杂。非单调推理过程就是建立假设，进行标准逻辑意义下的推理，若发现不一致，进行回溯，以便消除不一致，再建立新的假设。

1978 年 Reiter 首先提出了非单调推理方法封闭世界假设(CWA) [Reiter 1978]，并提出默认推理[Reiter 1980]。1979 年 Doyle 建立了非单调推理系统 TMS[Doyle 1979]。1980 年 McCarthy 提出限定逻辑[McCarthy 1980]。限定某个谓词 p 就是排除了以 p 的原有事实所建的大部分模型，而只剩下有关 p 的最小模型。不同形式的限定公理会引出不同类型的最小化标准。

定量模拟是计算机在科学计算领域的常规应用。但是人们常常不需要详细的计算数据，就能预测或解释一个系统的行为。这不能简单地通过演绎进行求解，人工智能提出定性推理的方法。定性推理把物理系统或物理过程细分为子系统或子过程，对于每个子系统或子过程以及它们之间的相互作用或影响都建立起结构描述，通过局部因果性的传播和行为合成，获得实际物理系统的行为描述和功能描述。最基本的定性推理方法有三种：即 de Kleer 的基于部件的定性方程方法，Forbus 的定性进程方法，Kuipers 的基于约束的定性仿真方法。定性与定量推理的结合将会对专家系统科学决策的发展产生重大影响。

在现实世界中存在大量不确定问题。不确定性来自人类的主观认识与客观实际之间存在差异。事物发生的随机性，人类知识的不完全、不可靠、不精确和不一致，自然语言中存在的模糊性和歧义性都反映了这种差异，都会带来不确定性。针对不同的不确定性的起因，人们提出了不同的理论和推理方法。在人工智能和知识工程中，有代表性的不确定性理论和推理方法有如下几种：

概率论被广泛地用于处理随机性以及人类知识的不可靠性。Bayes 理论被成功地用在 PROSPECTOR 专家系统中，但是，它要求给出假设的先验概率。在 MYCIN 中采用确信度方法是一种简单有效的方法。它采用了一些简单直观的证据合并规则。其缺点是缺乏良好的理论基础。

Dempster 和 Shafer 提出证据理论。该理论引进了信任函数的概念，对经典概率加以推广，规定信任函数满足概率公理更弱的公理，因此信任函数可以有作概率函数的超集。利用信任函数，人们无需给出具体的概率值，而只需要根据已有的领域知识就能对事件的概率分布加以约束。证据理论有坚实的理论基础，但是它的定义和计算过程比较复杂。近年来，

证据理论逐步引起人们的注意，出现了一些更加深入的研究成果和实用系统。例如，Zadeh 把证据理论的信任函数解释为二阶关系，并在关系数据库中找到了它的应用。Michacl 则把证据理论的基本概率函数推广到布尔代数。

1965 年 Zadeh 提出模糊集理论，以此为基础出现了一系列研究成果，主要有模糊逻辑、模糊决策和可能性理论。Zadeh 为了运用自然语言进行推理，对自然语言中的模糊概念进行了量化描述，提出了语言变量、语言值和可能性分布的概念，建立了可能性理论和近似推理方法，引起了许多人的研究兴趣。模糊数学已广泛应用于专家系统和智能控制中，人们还研制模糊计算机。我国学者在理论研究和应用方面均做了大量工作，引起国际学术界的关注。同时，这一领域仍然有许多理论问题没有解决，而且也存在不同的看法和争议，例如模糊数学的基础是什么？模糊逻辑的一致性和完全性问题。今后不确定推理的研究重点可能会集中在如下三个方面：一是解决现有处理不确定性的理论中存在的问题；二是大力研究人类高效、准确的识别能力和判断机制，开拓新的处理不确定性的理论和方法；三是探索可以综合处理多种不确定性的方法和技术。

证明定理是人类特殊的智能行为，不仅需要根据假设进行逻辑演绎，而且需要某些直觉技巧。机器定理证明就是把人证明定理的过程通过一套符号体系加以形式化，变成一系列能在计算机上自动实现的符号演算过程，也就是把具有智能特点的推理演绎过程机械化。中国科学院系统所吴文俊教授提出的平面几何及微分几何的判定法，得到国内外高度评价。

1.7 机器学习

知识、知识表示及运用知识的推理算法是人工智能的核心，而机器学习则是关键问题。数百年来，心理学家和哲学家们曾认为，学习的基本机制是设法把在一种情况下是成功的表现行为转移到另一类似的新情况中去。学习是获取知识、积累经验、改进性能、发现规律、适应环境的过程。图 1.2 给出了学习的简单模型。模型中包含学习系统的四个基本环节。环境提供外界信息，类似教师的角色。学习单元处理环境提供的信息，相当于各种学习算法。知识库中以某种知识表示形式存储信息。执行单元利用知识库中的知识来完成某种任务，并把执行中的情况回送给学习单元。学习使系统的性能得到改善。机器学习的研究可以使机器自动获取知识，赋予机器更多的智能。另一方面可以进一步揭示人类思维规律和学习奥秘，帮助人们提高学习效率。机器学习的研究还会对记忆存储模式、信息输入方式及计算机体系结构产生重大影响。

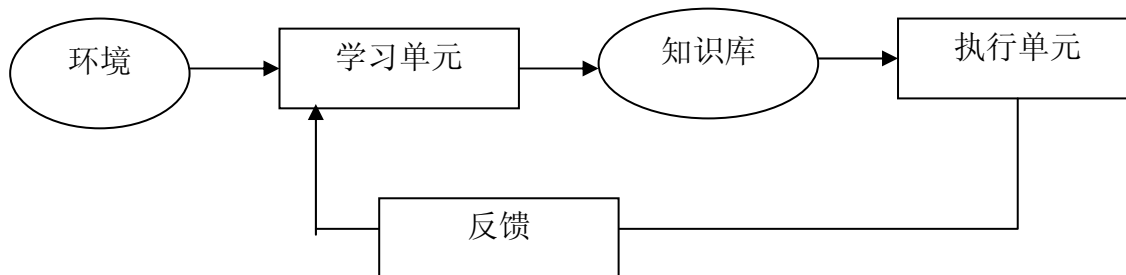


图 1.2 简单学习模型

机器学习的研究大致经历了四个阶段。早期研究是无知识的学习，主要研究神经元模型和基于决策论方法的自适应和自组织系统。但是神经元模型和决策论方法当时只取得非常有限的成功，局限性很大，研究热情大大降低。20 世纪 60 年代处于低潮，主要研究符号概念获取。1975 年 Winston 发表了从实例学习结构描述的文章，人们对机器学习的兴趣开始恢复，出现了许多有特色的学习算法。更重要的是人们普遍认识到，一个学习系统在没有知识的条件下是不可能学到高级概念的，因而把大量知识引入学习系统做为背景知识，使机器学习理论的研究出现了新的局面和希望。由于专家系统和问题求解系统的大量建造，知识获取成为严重的瓶颈，而这一问题的突破完全依赖于机器学习研究的进展。机器学习的研究开始进入新的高潮。

机器学习的风范主要有归纳学习、分析学习、发现学习、遗传学习和连接学习[Shi 1991]。过去对归纳学习研究最多，主要研究一般性概念的描述和概念聚类。提出了 AQ 算法、变型空间算法、ID3 算法等。类比学习是通过目标对象与源对象的相似性，从而运用源对象的求解方法来解决目标对象的问题。分析学习是在领域知识指导下进行实例学习，包括基于解释的学习、知识块学习等。基于解释的学习是从问题求解的一个具体过程中抽取出一般的原理，并使其在类似情况下也可利用。因为将学到的知识放进知识库，简化了中间的解释步骤，可以提高今后的解题效率。发现学习是根据实验数据或模型重新发现新的定律的方法。近年来，数据库知识发现引起人们极大的关注。从事人工智能研究和从事数据库研究的人们都认为这是一个极有应用意义的研究领域。数据库知识发现主要发现分类规则、特性规则、关联规则、差异规则、演化规则、异常规则等。数据库知识发现的方法主要有统计方法、机器学习、神经网络、多维数据库等方法。遗传学习起源于模拟生物繁衍的变异和达尔文的自然选择，把概念的各种变体当作物种的个体，根据客观功能测试概念的诱发变化和重组合并，决定哪种情况应在基因组合中予以保留。连接学习是神经网络通过典型实例的训练，识别输入模式的不同类别。

机器学习的研究尚处于初级阶段，必须大力开展研究的一个人工智能领域。只有机器学习的研究取得进展，人工智能和知识工程才会取得重大突破。今后机器学习的研究重点是研究学习过程的认知模型、机器学习的计算理论、新的学习算法、综合多种学习方法的机器学习系统等。

1.8 分布式人工智能

人们在研究人类智能行为中发现：大部分人类活动都涉及多个人构成的社会团体，大型复杂问题的求解需要多个专业人员或组织协作完成。“协作”是人类智能行为的主要表现形式之一，在人类社会中普遍存在。分布式人工智能（Distributed Artificial Intelligence，简称 DAI）正是为适应这种需要而兴起的。

自 20 世纪 80 年代以来，随着计算机网络、计算机通信和并发程序设计技术的发展，分布式人工智能逐渐成为人工智能领域的一个新的研究热点。分布式人工智能是人工智能的一个分支，它主要研究在逻辑上或物理上分散的智能动作者如何协调其智能行为，即协调它们的知识、技能和规划，求解单目标或多目标问题，为设计和建立大型复杂的智能系统或计算机支持协同工作提供有效途径。

DAI 一词产生于美国。第一届 DAI 会议“The Workshop on Distributed Artificial Intelligence”于 1980 年在美国的 Boston 的 MIT 召开。此后全世界各地的有关 DAI 或者包含 DAI 主题的各种会议不断举行，为 DAI 技术的发展和推广起了很大的促进作用。作为一门学科，DAI 的研究和实践不断地深入、扩大。随着新的基于计算机的信息系统、决策系统和知识系统在规模、范围和复杂程度上的增加，并且在这些系统中嵌入更加复杂的知识的要求的增加，DAI 技术的应用与开发越来越成为这些系统成功的关键。

一般，DAI 的研究可分为两个大方向：分布式问题求解（DPS: Distributed Problem Solving）和多主体系统（MAS: Multi-Agent System）。它们处于同一研究范系的两个端点上。DPS 的目标是要创建大粒度的协作群体，它们之间共同工作以对某一问题进行求解。在一个纯粹的 DPS 系统中，问题被分解成任务，并且为求解这些任务，需要仅为该问题设计一些专用的任务执行系统。所有的交互（如果有，如协作等）策略都被集成为系统设计的整体部分。这是一种从顶向下设计的系统，因为处理系统是为满足在顶部所给定的需求而设计的。

与 DPS 系统相反，在一个纯粹的多主体系统中，主体是自主的，可能是预先存在的，并且是异构的。多主体系统并不限制为一个单一的任务。多主体系统的研究涉及到在一组自主的智能主体之间协调其智能行为，协调它们的知识、目标及规划等以便联合起来采取行动或求解问题。虽然在这里一个主体也可以是某个任务的执行者，但它具有“开放的”接口，任何“人”都可以对其进行存取。该主体不仅可以处理单一目标，而且可以处理不同的多个目标。

目前计算机的应用越来越广泛，所需处理的问题越来越复杂，问题求解所涉及的信息、资料、数据很难集中式地处理，并且求解过程也难以集中控制。这种数据或知识的分布以及并发处理，对 AI 发展带来巨大潜力的同时也带来了各种有待于解决的困难问题。DAI 系统中各主体在空间上的分布性，时间上的并发性以及在逻辑上的依赖关系使得多主体系统的求解行为较之单主体系统更复杂。

对 DAI 系统的研究主要原因可概括为：

(1) 技术基础——处理器硬件结构技术及处理器之间的通信技术的进步使得大量复杂的并且是异步执行的处理器之间的互联成为可能。这种联结可以是基于共享或分布式内存的紧耦合的系统，也可以是基于局域网络的比较松耦合的系统，甚至可以是基于地理上分布的通信网络的非常松散的耦合系统。

(2) 分布式问题求解——很多的人工智能应用在本质上都是分布的。这些应用可能是空间分布的，如对空间上分布的传感器的数据的解释和集成，或者是对工厂中共同工作的机器人的控制；这些应用也可能是功能分布的，如为了解决复杂的病例将几个专业的医学诊断系统联合起来；这些应用还可能是时序上分布的，例如在一个工厂中，生产线是由几个工序组成，每个工序都由一个专家系统进行调度。

(3) 易于系统集成——分布式人工智能系统支持模块性的设计及实现。把一个复杂的系统分解成一些相对来说简单的、处理某个特定问题的模块，使得系统便于建造、调试及维护。对多个模块的硬件或软件错误的处理比单一的整体模块具有更大的灵活性。另一方面，大量的已经存在的集中式的人工智能应用系统，如果可以稍加修改即用于构成分布式人工智能系统，则可以产生很大的经济效益和社会效益。例如，原来的肝病诊断专家系统、胃病诊断专家系统、肠道疾病诊断专家系统等独立的系统，如果可以通过少量修改建成消化道疾病诊断多专家系统则可以节省大量开发时间，并产生更大的效用。我们提出的插件式构造主体的方法，就是一种有效地集成已有人工智能系统的方法。

(4) 智能行为的新途径——通过智能主体实现自主的智能行为。要使人工智能系统成为思维社会的组成，它必须具有与环境之间进行交互的作用，以及彼此协作和协调的能力。

(5) 认识论上的意义——分布式人工智能可用来研究和验证社会学、心理学、管理学等问题和理论。通过信念、知识、希望、意图、承诺、注意、目标、协作等，实现协同工作的多主体系统，为理解和仿真认识论问题提供有效的手段。

所以，无论从技术上还是社会需求上，DAI 系统的出现与发展都是必然的。利用 DAI 技术来解决大型的军事领域的问题是必要的，也是十分自然的。目前，国内从事这方面的研究已取得了一定的成果。

多主体系统是分布式人工智能研究的一个分支。在多主体系统中，主体是一个自主的实体，它不断地与环境发生交互作用。同时在该环境中还有其他的进程发生，也存在其他的主体。或者说，主体是一个其状态由心智部件，如信念、能力、选择、意图等组成的实体。在一个系统中，主体可以是同构的，也可以是异构的。多主体的研究涉及到在一组自主的智能主体之间协调其智能行为，协调它们的知识、目标、意图及规划以联合起来采取行动或求解问题。主体之间可能是协作关系，也可能存在着竞争。分布式人工智能和多主体系统的一个共同特点就是分布式的实体行为。多主体系统可看作是采用由底向上的设计方法设计的系统。因为在原理上，分散自主的主体首先被定义，然后研究怎样完成个人或几个实体的任务求解。多主体系统不仅可以处理单一目标，也可以处理不同的多个目标。多主体系统主要研究在逻辑上或物理上分离的多个主体如何并发计算、相互协作地实现问题求解。其主要目的在于分析

和设计大型复杂的协作智能系统，如大型知识信息系统、智能机器人等。

目前，多主体系统的研究非常活跃。多主体系统试图用主体来模拟人的理性行为，主要应用在对现实世界和社会的模拟、机器人和智能机械等领域。主体本身需要具有自治性、对环境的交互性、协作性、可通讯性，以及长寿性、自适应性、实时性等特性。而在现实世界中生存、工作的主体，要面对的是一个不断变化的环境。在这样的环境中，主体不仅要保持对紧急情况的及时反应，还要使用一定的策略对中短期的行为作出规划，进而通过对世界和其它主体的建模分析来预测未来的状态，以及通过通讯语言实现和其他主体的协作或协商。为了使主体表现出这样的性质，需要研究主体的结构。因为主体的结构和它的功能是紧密相关的，不合理的结构将大大限制主体的功能，而合理的结构则将给实现主体的高度智能化提供支持。我们提出一种复合式的结构，即在一个主体中有机地组合了多种并行执行、相对独立但又相互作用的智能形态，其中包括反射、规划、建模、通讯和决策生成等。我们提出并部分实现了一个多主体处理环境 MAPE (Multi-Agent Processing Environment)。在 MAPE 中，我们提出了一种基于主体内核的插件式地构造主体的方法。使用 MAPE 环境和插件式的构造方法，可以方便地构造和调试复合式结构的主体。

1.9 人工思维模型

计算机的发展已经经历了两个阶段。在第一个阶段，采用冯·诺伊曼体系结构，主要用于数值计算、文档处理、数据库管理和检索。这些应用都有明确的算法，仅在编程方面较困难。第二个阶段是面向符号和逻辑的处理，主要通过推理，进行知识信息处理。如何确定有效的算法是研究的重点。上述这些应用，都属于理想世界的问题，具有良定义和明确的描述。现实世界的问题大多具有病态定义的结构，例如模式识别、不完全信息的问题求解和学习等，而这些问题属于直觉信息处理的范畴。

为了处理直觉信息，必须研究柔性信息处理的理论和技术。真实世界的所谓柔性问题具有下列特点：

- (1) 包含意义不明确或不确定信息的各种复杂情况的集成；
- (2) 主动获取必要的信息和知识，通过归纳学习范化知识；
- (3) 系统本身能适应用户和环境的变化；
- (4) 根据处理对象系统进行自组织；
- (5) 容错处理能力。

实际上，具有大规模并行和分布式信息处理功能的人脑神经网络自然地支持柔性信息处理。由此，作者提出了人工思维模型(见图 1.3)。

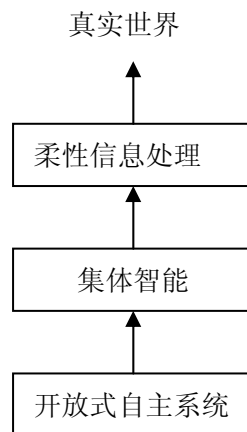


图 1.3 人工思维模型

图 1.3 所示的人工思维模型清楚地表明，人工思维将以开放式自主系统为基础，充分发挥各种处理范型的特长，实现集体智能，才能达到柔性信息处理，解决真实世界的问题。

1.10 知识系统

人工智能研究的一个最重要的动力是建立知识系统以求解困难问题。20 世纪 80 年代以来，知识工程成为人工智能应用最显著的特点。知识系统包括专家系统、知识库系统、智能决策系统等。1965 年，为阐明有机化学结构而创建的 DENDRAL 发展成为一类专家系统的程序。这类计算机程序包括两部分：一是知识库，它表示和存储由任务所指定领域知识的一组数据结构集合。知识库不仅包含了有关领域的事实，而且包含专家水平的启发式知识。另一是推理机，它是构造推理路径的一组推理方法集合，以便导致问题求解、假设的形成、目标的满足等。由于推理采用的机理、概念不同，推理机形成多种范型的格局。

知识库系统是把知识以一定的结构存入计算机，进行知识的管理和问题求解，实现知识的共享。美国推出了 KBMS 软件产品。日本的 NTT 公司也研制成了 KBMS。中国科学院计算所于 1990 年完成了国家七五重点科技攻关项目知识库管理系统软件 KBMS。这些软件的明显特色是将推理和查询结合起来，改善了知识库的维护功能，为开发具体领域的知识系统提供有用的环境。

决策支持系统（DSS）是在管理信息系统（MIS）的基础上发展起来的，这一概念始于 20 世纪 70 年代初。由于它是提高企业竞争力、生产力以及决定经营成败的重要工具，所以发展很快。在国外，已为各级决策人员所采用。在国内也引起了各方面的关注。决策支持技术是支持科学决策的关键技术之一。早期的决策支持系统是在管理信息系统的基础上，增加一些规范模型(如运筹学模型、经济计量模型等)而成。1980 年，Sprague 提出基于数据库和模型库的 DSS 结构，产生了很大的影响。最近几年，人工智能技术逐步应用于 DSS，产生了智能化的 DSS。1986 年，作者提出由数据库、模型库、知识库等组成的智能决策系统(史忠植 1988)，为解决半结构、非结构化的决策问题提供了有效的手段，提高了科学管理的水平。智

能决策系统的特点是将人工智能技术应用于 DSS, 并且将数据库技术、情报检索技术与基于模型和方法的定量分析技术相结合。20 世纪 90 年代, 我们采用多主体技术建立群体决策系统, 引起了人们的兴趣。

建造智能系统可以模仿、延伸和扩展人的智能, 实现某些“机器思维”, 具有极大的理论意义和实用价值。根据智能系统具有的知识和处理范型的情况, 可以分成四类:

单领域知识单处理范型智能系统、多领域知识单处理范型智能系统、单领域知识多处理范型智能系统、多领域知识多处理范型智能系统。

1. 单领域知识单处理范型智能系统

系统具有单一领域的知识, 并且只有一种处理范型。例如, 第一代、第二代专家系统和智能控制系统属于这种类型。

专家系统是运用特定领域的专门知识, 通过推理来模拟通常由人类专家才能解决的各种复杂的、具体的问题, 达到与专家具有同等解决问题能力的计算机智能程序系统。它能对决策的过程做出解释, 并有学习功能, 即能自动增长解决所需的知识。第一代专家系统(DENDRAL、MACSYMA 等)以高度专业化、求解专门问题的能力为特点, 但在体系结构的完整性、可移植性等方面存在缺陷, 求解问题的能力弱。第二代专家系统(MYCIN、CASNET、PROSPECTOR、HEARSAY 等)属单学科专业型、应用型系统, 其体系结构较完整, 移植性方面也有所改善, 而且在系统的人机接口、解释机制、知识获取技术、不确定推理技术、增强专家系统的知识表示和推理方法的启发性、通用性等方面都有所改进。

2. 多领域知识单处理范型智能系统

多领域知识单处理范型智能系统具有多种领域的知识, 而处理范型只有一种。大多数分布式问题求解系统、多专家系统属于这种类型。一般采用专家系统开发工具和环境来研制这种大型综合智能系统。

由于智能系统在工程技术、社会经济、国防建设、生态环境等各个领域的广泛应用, 对智能系统的功能提出多方面的要求。许多实际问题的求解, 例如, 医学诊治、经济计划、军事指挥、金融工程、作物栽培、环境保护等, 往往需要应用多学科、多专业的专家知识和经验。现有的许多专家系统大多数是单学科、专门性的小型专家系统, 不能满足用户的实际需求。建立多领域知识单处理范型智能系统在一定程度上可以达到用户的要求。这类智能系统的特点是:

- (1) 面向用户实际的复杂问题求解;
- (2) 应用多学科、多专业、多专家的知识和经验, 进行并行协同求解;
- (3) 基于分布式、开放性软硬件和网络环境;
- (4) 利用专家系统开发工具和环境;
- (5) 实现知识共享与知识重用。

3. 单领域知识多处理范型智能系统

单领域知识多处理范型智能系统具有单一领域的知识, 而处理范型有多种。例如, 混合

智能系统属于这种类型。一般可以用神经网络通过训练，获得知识。然后，转换成产生式规则，提供给推理机在求解问题时使用。

在进行问题求解时，也可以采用多种机制处理同一个问题。例如，疾病诊断系统，既可采符号推理的方法，也可通过人工神经网络。让它们同时处理相同的问题，然后比较它们的结果，这样容易取得正确的结果，避免片面性。

4. 多领域知识多处理范型智能系统

图 1.4 给出了多领域知识多处理范型智能系统的示意图。该种系统具有多种领域的知识，而且处理范型也有多种。图中集体智能(collective intelligence)的含义是, 在多种处理范型的环境下，各种处理机制各行其事，各司其职，协调工作，表现为集体的智能行为。

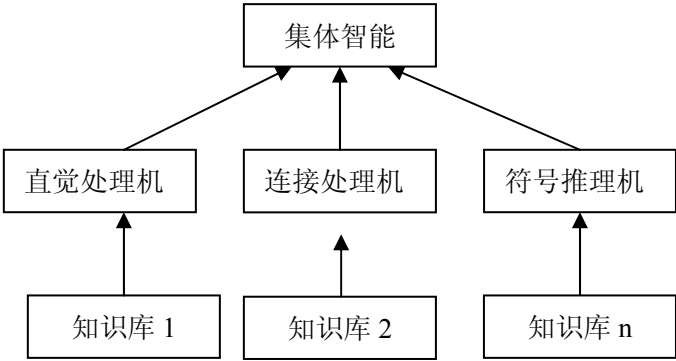


图 1.4 多领域知识多处理范型智能系统

综合决策系统、综合知识系统属于多领域知识多处理范型智能系统。在这种系统中，基于推理的抽象思维采用符号处理的方法；而基于模式识别、图像处理之类的形象思维采用神经计算。

智能问题多数具有病态结构，研究的对象也在不断地变化，很难找到一种精确的算法进行求解。构造人机统一、与环境进行交互、反馈的开放系统是解决这类智能问题的途径。所谓开放系统是指系统在操作过程中永远有未预料的后果，并能在任何时候从外部接收新的信息。

在总结和分析已有知识系统的设计方法和实现技术的基础上，人们研究智能主体技术，用来实现多种知识表示、综合知识库、自组织协同工作、自动知识获取、不断改善系统的智能行为等功能的大型综合知识系统。这类知识系统是当前实现多领域知识多处理范型智能系统的主要途径。

习 题

1. 什么是人工智能？它的研究目标是什么？
2. 简述人工智能研究发展的主要阶段。

3. 人工智能研究的基本问题是什么？
4. 什么是物理符号系统？什么是物理符号假设？
5. 什么是符号智能？什么是计算智能？
6. 请描绘机器学习的简单模型，并论述各个基本环节的基本内容。
7. 什么是分布式人工智能，它包括哪些研究方向。
8. 参考相关文献，讨论目前的计算机是否可以解决下列任务：
 - a) 在国际象棋比赛中战胜国际特级大师
 - b) 在围棋比赛中战胜九段高手
 - c) 发现并证明新的数学定理
 - d) 自动找到程序中的 bug
9. 知识系统如何分类？怎样建立集体的智能行为？

第二章 人工智能逻辑

2.1 概述

亚里斯多德从数学的研究中分离出逻辑学。莱布尼茨把数学的方法引入逻辑领域，创立了数理逻辑。在本世纪 30 年代以后，数学方法广泛渗透与运用于数理逻辑，使得数理逻辑成为数学领域中与代数、几何等并列的学科之一。现代数理逻辑可以分为逻辑运算、证明论、公理集合论、递归论和模型论。

逻辑方法是计算机科学，尤其是人工智能研究中的主要形式化工具。从语义学、程序设计语言、程序规范理论到程序验证理论，从数据库理论、知识库理论到智能系统，直到机器人的研究，所有的领域或多或少都与逻辑学有着若干联系。而逻辑方法之所以成为计算机科学和人工智能研究的主要工具，其根源可以追溯到计算机科学和逻辑学所追求的目标在深层次上的一致性。从本质上来说，计算机科学就是要用计算机来模拟人脑的行为和功能，使计算机成为人脑的延伸。而对于人脑的行为和功能的模拟，实质上就是模拟人的思维过程。正是计算机科学所追求的这个目标，逻辑学这个以研究人的思维规律和法则的学科，它的研究方法和研究成果自然而然地成为计算机科学研究所选用的工具。由于人类智能行为在很大程度上是通过语言和文字表达出来的，因此，从技术上来说，计算机科学模拟人类思维，也是从模拟人类的自然语言作为出发点的。围绕语言的概念进行的研究是计算机科学（尤其是人工智能）的一个核心领域。

逻辑学研究人的思维是从研究人的自然语言开始入手的，计算机科学模拟人的思维同样是从语言开始的。与语言相关的论题是贯穿计算机科学领域的重要问题，许多的领域与语言相关，例如，软件领域的程序设计语言和形式语义学，人工智能领域中数据和知识的表示和推理，自然语言处理中的计算语言学等等。总起来说，表示和推理是计算机科学和人工智能领域的基本问题。大多数智能行为的高级形态需要对于知识进行直接的表示，而形式逻辑是知识表示的重要方式。

智能行为的基础是知识，尤其是所谓的常识性知识。人类的智能行为对于知识的依赖主要表现在对于知识的利用，即利用已经具有的知识进行分析、猜测、判断、预测等等。人类利用知识可以预测未来，由已知的情况推测未知的情况、由发生的事件预测还未发生的事件等等。但是，当人们希望计算机具有智能行为时，除了告诉计算机如何像人一样地利用知识以外（对于知识进行推理），一个更为基础和先行的工作是如何使计算机具有知识（对于知识进行表示），即在计算机上如何表达人类的知识。要使得一个系统成为智能的系统，它必须具有知识，而且目前能够达到这个目的的唯一方法就是把系统嵌入到某种所谓的知识表示的结构中去。由此分析以及通过对比我们不难看到，人工智能对于知识的这两个关注点与逻辑对

于自然语言的两个关注点（自然语言的精确结构和推理问题）是完全重合的。逻辑学对于自然语言的精确结构以及对于推理的研究正好服务于人工智能对于知识研究的两点要求。当人们利用逻辑于智能系统时，这两个方面是同时加以考虑的。由于一个逻辑系统的表示能力强弱和推理性质优劣之间存在某种冲突，因此，在实际的研究中常常在它们两者之间进行平衡和折衷。从某种意义上来说，人们利用逻辑对于知识进行推理在逻辑应用于人工智能研究中的作用远远超过了单纯的知识表示。因为，逻辑推理所涉及的不单单是知识的外在形式，它事实上涉及到的是知识的内容，是形式的知识所包含的意义的内在联系。虽然从表面上看，推理是形式的，但是逻辑学所追求和保证的就是知识的形式和内容之间的固定联系。逻辑学的这种追求与保证使得对于知识的推理使得机器对于知识的理解不是形式上的而是含义上的。研究包含在知识的形式表示之上的知识的含义，对于智能研究是本质的一点，这也正如研究信息的含义高于信息的形式一样。比通信更复杂的信息活动（如推理、思维和决策等等）恰恰需要利用信息的内容和价值因素。一般来说，不了解信息的内容和价值，很难作出科学的推理、思维和决策。

多数的基于逻辑的智能系统使用一阶逻辑或者它的一些扩张形式。一阶逻辑的优点是它具有相当强的表达能力。有的人工智能专家坚信所有的人工智能中的知识表示问题完全可以在一阶逻辑的框架中得以实现。一阶逻辑在表达不确定性知识时其表达能力也是很强的。例如， $\exists x P(x)$ 表达在所考虑的论域中存在一个具有性质 P 的对象，而具体的是哪一个对象具有此性质则是待确定的；再如， $P \vee Q$ 表示 P 和 Q 这两个性质之间有一个是成立的，至于到底是哪一个成立则是根据具体的情况而定的。此外，一阶逻辑还有一个完备的公理系统。完备的公理体系为我们设计有关推理的策略和算法提供了一个参考基准。虽然，有人坚信从本质上看，一阶逻辑对于知识表示是足够的，但从实际应用的角度看，为方便、清楚和简洁起见，知识表示不一定非得从一阶逻辑出发。事实上，人们从实际应用出发已经发明和建立了许多适用于不同目的的逻辑系统。以下是其中的一些例子：

(1) 为了表示关于认知的有关概念，如相信、知道、愿望、意图、目标、承诺等等，人们引进了刻画各种认知概念的模态逻辑；

(2) 为了刻画智能系统中的时间因素，人们在逻辑系统中引进时间的概念，提出了各种时序逻辑；

(3) 为了描述各种不确定的和不精确的概念，人们引进了所谓模糊逻辑；模糊逻辑是直接建立在自然语言上的逻辑系统，与其它逻辑系统相比较，它考虑了更多的自然语言的成分。按照其创始人 Zadeh 的说法就是词语上的计算，表示为一个公式，即 fuzzy logic=computing with words；

(4) 人类的知识与人类的活动是息息相关的，人类正是在各种活动和行为中获得知识的。因此，行为或者动作的概念在智能系统中是一个关键的概念。动作的概念与一般逻辑中的静态的概念很不相同，它是一个动态的概念，动作的发生影响着智能系统的性质。对于动作的考虑，给人工智能界带来了许多难题，如框架问题、量词问题等等。为了刻画动作的概念，

人们引进了一些新的逻辑体系来刻画它。

(5) 计算机对于人类进行决策时进行若干方面的支持已经成为计算机应用的一个重要方面。人类在决策时，对于各种方案和目标有一定的偏好和选择。这时“偏爱”就成为了一个基本的概念。为了表述和模拟人类在决策时的选择的规律和行为，对于“偏爱”这个词的研究就是不可避免的。于是，基于管理科学的所谓的偏爱逻辑被提出并加以研究。

(6) 时间是智能系统中最重要几个概念之一。人类使用各类副词来对时间概念加以描述。例如，“一会儿”“相当长”“断断续续地”“偶尔”等等，这一类词在我们的日常生活中比比皆是。含有这些词的句子显然是很难用经典的时序逻辑来刻划的，于是有人引进了一种逻辑系统专门刻划这类句子。其基本思想是利用数学中积分的思想，通过对时间的某种像积分那样的表示和运算来形式化这些句子。

2.2 逻辑程序设计

Prolog 程序就是一种逻辑程序，后面我们的讨论基于的都是 Prolog 语言。Prolog 是一种交互式语言，是人们多年研究成果的结晶。Prolog 的第一个正式版本是 1970 年代法国 Marseilles 大学的 Alain Colmerauer 作为 PROgramming in LOGic 的工具开发出来的。今天 Prolog 已成为人工智能应用和专家系统开发的重要的工具。

Prolog 是一种描述性语言，只要给定所需的事实和规则，Prolog 使用演绎推理方法就可对问题进行求解。即只需告诉计算机“做什么”，至于“怎么做”则是由 Prolog 自动完成的。除此之外，Prolog 还有下面一些特点：

1. Prolog 是数据和程序的统一。Prolog 提供了一种一致的数据结构：项。所有数据和程序都是由项构造而成的。在智能程序中常需要将一段程序的输出数据作为新产生的程序来执行，因此人工智能语言应具有数据和程序结构一致的特性。
2. Prolog 能够自动实现模式匹配和回溯。这些是人工智能系统中最常用的、最基本的操作。
3. Prolog 具有递归的特点，它反映在 Prolog 程序和数据结构的设计中。由于这一特性，一个大的数据结构常常可以用一个小的程序来处理。一般情况下，对一个应用来说，用 Prolog 语言写的程序长度是用 C++语言写的程序长度的十分之一[119]。

Prolog 的所有这些特性，使得 Prolog 特别适用于描述智能程序，适用于自然语言处理、定理证明和专家系统等等。

2.2.1 逻辑程序定义

在定义逻辑程序之前，首先定义 Horn 子句，它是逻辑程序的组成要素。

一个子句由两部分组成：头部和体。IF-THEN 规则的结论称为头部，前提部分称为体。

则 Horn 子句可以定义为:

定义 2.1 Horn 子句是头部最多包含一个文字（命题或谓词）的子句。

Horn 子句在 Prolog 中有三种表示形式:

- (1) 无条件子句 (事实): A .
- (2) 条件子句(规则): $A:-B_1,\dots,B_n$.
- (3) 目标子句(问题): $?-B_1,\dots,B_n$.

上述三种 Horn 子句均具有明显的非形式语义:

- (1) 无条件子句 A : 表示对变量的任何赋值, A 均为真。
- (2) 条件子句 $A:-B_1,\dots,B_n$: 表示对变量的任何赋值, 如果 B_1,\dots,B_n 均为真, 则 A 为真。
- (3) 目标子句 $?-B_1,\dots,B_n$: 其逻辑形式为

$$\forall x_1 \dots x_n (\neg B_1 \vee \dots \vee \neg B_n),$$

等价于 $\neg \exists x_1 \dots x_n (B_1 \wedge \dots \wedge B_n)$ 。它视作推理的目标。

例如, 对于下面两个 Horn 子句:

- i) $W(X,Y) :- P(X), Q(Y)$.
- ii) $?-R(X,Y), Q(Y)$.

在 i) 中 $W(X,Y)$ 为头, $P(X), Q(Y)$ 为体。在 ii) 中 $R(X,Y), Q(Y)$ 为体, 头为空子句。事实上, ii) 表示一个询问, $R(X,Y), Q(Y)$ 是否为真, 或者 X 和 Y 取什么值的时候, $R(X,Y) \wedge Q(Y)$ 为真。

定义 2.2 逻辑程序就是由 Horn 子句构成的程序。在逻辑程序中, 头部具有相同谓词符的那些子句称为该谓词的定义。

例如下面两个谓词逻辑句子, 每个句子都只有一个头。

$Father(X,Y) :- Child(Y,X), Male(X)$.

$Son(Y,X) :- Child(Y,X), Male(Y)$.

上述两个子句都是 Horn 子句, 因此它们构成一个逻辑程序。假设还有下面三个事实子句:

$Child(xiao-li, lao-li)$.

$Male(xiao-li)$.

$Male(lao-li)$.

如果把上述规则和事实加入 Prolog 中, 编译执行后, 给出下面的查询, 则有:

1. 目标: $?- Father(X,Y)$.
则会得到: $Father(lao-li, xiao-li)$.
2. 目标: $?- Son(Y,X)$.
则会得到: $Son(xiao-li, lao-li)$.

上面我们已经完成了对 Prolog 语言简单的描述。

2.2.2 Prolog 数据结构和递归

在非数值程序设计中, 递归是十分重要的工具。在 Prolog 中, 递归也是其重要的特性, 这反映在 Prolog 的数据结构和程序中。

Prolog 提供了一个一致的数据结构称为项。所有的数据和 Prolog 程序都是由项构造而成的。Prolog 的项可以定义为:

$\langle \text{项} \rangle ::= \langle \text{常量} \rangle | \langle \text{变量} \rangle | \langle \text{结构} \rangle | (\langle \text{项} \rangle)$

结构称为复合项, 它是由一组其他对象 (也可以为结构) 组成的单个对象。

$\langle \text{结构} \rangle ::= \langle \text{函数符} \rangle (\langle \text{项} \rangle \{, \langle \text{项} \rangle \})$

<函数符>::=<原子>

Prolog 的结构相当于通常程序设计语言中的结构和记录。为了灵活地描述序列、集合这样的对象，Prolog 提供了表这一数据结构。表是非数值程序设计中一种最常用的数据结构。在 Prolog 中，一个表的元素可以是原子、结构或任何其他项，包括表，因此表是递归定义的。表可以表示成一个特殊的二元函数 `cons(X,Y)`，X 是表头，Y 是表尾（即表中除去表头剩下部分组成的表），它是 Prolog 中最重要的数据结构。Prolog 采用下面的记号表示表：

<code>[]</code> 或 <code>nil</code>	空表
<code>[a]</code>	<code>cons(a,nil)</code>
<code>[a,b]</code>	<code>cons(a,cons(b,nil))</code>
<code>[a,b,c]</code>	<code>cons(a,cons(b,cons(c,nil)))</code>
<code>[X Y]</code>	<code>cons(X,Y)</code>
<code>[a,b c]</code>	<code>cons(a,cons(b,c))</code>

Prolog 的递归性还体现在程序中。例如对于内部谓词 `member`，用于确定一个元素是否为一给定表的成员。`member` 谓词定义如下：

`member(X, [X | _]).`

`member(X, [_ | Y]) :- member(X,Y).`

上面在定义谓词 `member` 时又调用了它自己，因此 `member` 是递归定义的。

上述这种递归模式在 Prolog 中会大量的出现。

2.2.3 SLD 归结

在讨论 SLD（有选择的线性）归结之前，首先给出下面的定义。

定义 2.3 定程序子句是下述形式的子句：

$A :- B_1, B_2, \dots, B_n$

其中在头部（结论）只有一个正文字（即 A），而体中有零个、一个或多个文字。

定义 2.4 定程序是定程序子句的有限集合。

定义 2.5 定目标是下述形式的子句：

$? :- B_1, B_2, \dots, B_n$

其中头部（结论）是为空的子句。

定义 2.6 SLD 归结代表用于定子句的具有选择函数的线性归结。

如果用 P 表示程序，G 表示目标，则逻辑程序的求解过程就是寻找 $P \cup \{G\}$ 的 SLD 归结的过程。为了确定一个归结过程，需要给出如何选择子目标的计算规则，以及程序空间的搜索规则。在理论上，人工智能中对状态空间的任何搜索策略都可以使用。但是对于 Prolog 来说，实现效率的考虑是放在第一位的。标准的 Prolog 的 SLD 归结过程可归纳如下：

- （1） 计算规则总是选取最左边的子目标：从左向右
- （2） 搜索规则是深度优先+回溯。
- （3） 按 P 中子句的书写顺序进行尝试：从上到下
- （4） 省去了合一算法中的 `occur` 检查。

上述四条限制会得到下面的问题。

1、深度优先策略有简单高效的实现方法。

这是采用这一搜索策略的根本原因，原则上可以采用一个“目标栈”就可以实现它（见第二章的深度优先搜索）。目标栈在任何时候总代表 SLD 树上正在被搜索的分枝。搜索过程由一系列进栈、出栈操作组成。当栈顶的子目标与 P 中某子句头部合一成功后，归结式进栈。如没有可以合一的子句，则执行回溯操作，即栈顶元素出栈，为新的栈顶尝试下一个可合一子句。如此等等。

例 2.1 设程序为：

p(X, Z) :- q(X,Y), p(Y,Z).
p(X,X).
q(a, b).

目标子句为：?-p(X,b).

则标准 Prolog 的目标栈为：

?-p(X,b).				G 进栈，开始
?-p(X,b).	?- q(X,Y), p(Y,b).			归结式进栈
?-p(X,b).	?- q(X,Y), p(Y,b).	?- p(b, b).		归结式进栈
?-p(X,b).	?- q(X,Y), p(Y,b).	?- p(b, b).	?-q(b,W), p(W,b)	归结式进栈
?-p(X,b).	?- q(X,Y), p(Y,b).	?- p(b, b).	□	出栈，新归结式□进栈
?-p(X,b).				出栈（三次）
□				新归结式□进栈
				出栈，结束

2、深度优先搜索规则破坏了完备性

深度优先是不完备的，这一问题是不可能通过重新排列子目标序列和重排子句序列来完全克服的，但是可以解决一定的问题。例如程序

(1) p(f(X)):- p(X).
(2) p(a).

和目标子句?-p(Y)，在运行时会陷入死循环。而重新排列子句（1）（2）的顺序后，就会有 Y=a,Y=f(a),...。

考虑另一程序：

(1) q(f(X)) :- q(X).
(2) q(a).
(3) r(a).

目标子句 G: ?-q(Y), r(Y)。在运行时陷入死循环。但改变 G 中子目标的顺序，G: ?- r(Y),q(Y)，就可以得到唯一解：Y=a。

但是上述两种方法都不能从根本上解决问题。要保证完整性，Prolog 的搜索规则中必须包含某种宽度优先的成分。但这必然降低系统的时空效率，且实现难度增大。另一种可行的方法是保持 Prolog 的深度优先搜索策略，必要时用 Prolog 语言本身编写实现其他搜索策略的程序。

3、省略 occur 检查破坏了 SLD 归结法的正确性。

我们知道，合一算法中的 occur 检查是极费时间的。具有 occur 检查时，每次合一需要线性长度的时间，所以对于谓词 append 需要 $O(n^2)$ 时间，这里 n 是表的长度。由于实际的 Prolog 程序很少碰见确实需要做 occur 检查的合一运算，大多数 Prolog 系统都在合一算法中省去了 occur 检查。

但是省略了 occur 检查破坏了 SLD 归结法的正确性：在确实发生变量出现在项的内部时，

本应得到不可合一的结论。但省略了 occur 检查, 则得出可合一的结论, 且给出变量的“循环约束”, 这显然是错误的推理。

例如, 设程序 P 为:

$p(Y, f(Y)), G \text{ 为 } ?-p(X, X)$ 。

则合一算法在为 $\{p(X, X), p(Y, f(Y))\}$ 寻找最一般的合一置换时, 若省去了 occur 检查, 将会得到 $\theta = \{Y/X, f(Y)/Y\}$ 这样错误的置换。若后面的 SLD 归结过程不用到变量 Y, 这一错误将被掩盖起来, 若后面又用到 Y, 则 Prolog 将陷入死循环。

2.2.4 非逻辑成分: CUT

程序是算法的体现, 逻辑程序设计的基本公式是:

算法 = 逻辑 + 控制

其中逻辑部分说明“干什么”, 控制部分说明“怎么干”。程序员只需给出逻辑部分, 控制部分应由逻辑程序设计系统自动处理。可惜, 现今的 Prolog 系统一般达不到这一点。如前所述, 为了程序的正确运行, 程序员还需要考虑子句的次序等。另外, 由于 Prolog 运行控制系统使用深度优先搜索策略, 则可能产生某个无穷的分枝。因为 Prolog 深度优先搜索是基于栈实现的, 则无限的 SLD 树将会产生栈溢出。这样就无法得到任何解答。这种问题可以通过在程序中适当地放置“CUT”语句解决。

从说明性语义来看, CUT 是一种非逻辑成分的控制机制, 用“!”表示, 可以用在条件子句的体内, 或目标子句中, 并把它看做是一个恒真的原子。这样, 一个程序和目标中可以任意插入“!”而不影响程序的说明性语义。

但是从过程语义来看, CUT 传递了一定的控制信息。设目标子句 G 为:

$?-A_1, \dots, A_{m-1}, A_m, A_{m+1}, \dots, A_k$

程序子句 C 为:

$A_i :- B_1, \dots, B_i, !, B_{i+1}, \dots, B_q$

标准的 Prolog 在求出 A_1, \dots, A_{m-1} 的一个解后得到目标 G', 它先解 A_m , 若 A_m 可与 A 合一, 则 C 的体经过变量置换后成为新的目标子句的一部分, 称 G' 是包含! 的子句 C 的父目标, A_m 称为截断点。当轮到解! 时, 它总是立即成功。但当! 后面的子目标无解而引起回溯(或为了求 G 的所有解而回溯)时, 从截断点 A_m 到! 的所有其他解都不被考虑, Prolog 接着求解 A_m 前的那个子目标 A_{m-1} 的其他解。这就是! 传递的控制信息, 在 SLD 树上, “!”剪去了以它的父目标为根的那个子树中尚未被搜索的部分。

例如, 设程序 P 为:

- (1) $p(a).$
- (2) $p(b).$
- (3) $q(b).$
- (4) $r(X) :- p(X), q(X).$
- (5) $r(c).$

子目标 G 为: $?-r(X)$ 。标准 Prolog 系统建立 SLD 树如图 2.1。若 P 的 (4) 中插入! 变成:

(4)' $r(X) :- p(X), !, q(X).$

则相应的 SLD 树如图 2.2 所示, 它被剪去了一部分, 从而无解。

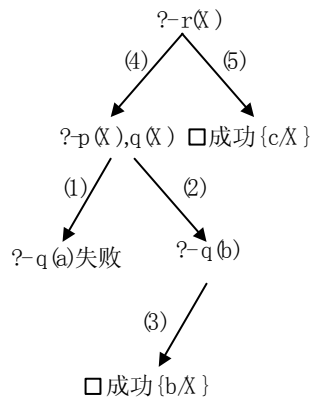


图 2.1 无! 的 SLD 归结树

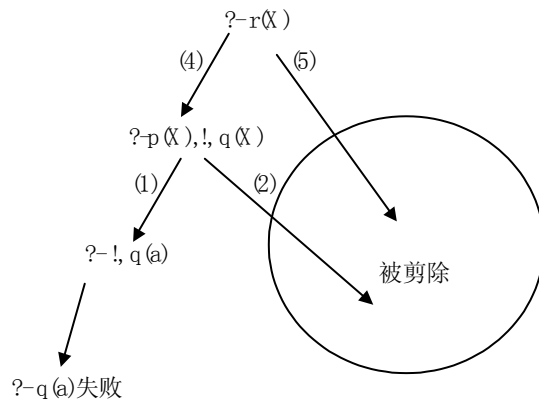


图 2.2 有! 的 SLD 归结树

从上面可以看到，使用 CUT 具有一定的风险，使用不当的话有可能删除成功的分枝，破坏了 SLD 归结法的完备性。但是更为严重的是 CUT 破坏了逻辑程序的两种语义的一致性。考虑求两个数最大值的程序 P:

$\text{max}(X, Y, Y) :- X \leq Y.$

$\text{max}(X, Y, X) :- X > Y.$

P 的说明性语义和过程性语义是一致的。若增加 CUT 谓词，得到如下的程序 P1:

$\text{max}(X, Y, Y) :- X \leq Y, !.$

$\text{max}(X, Y, X) :- X > Y.$

P1 的说明性语义和过程性语义都没有改变，只是运行效率有所提高。为了进一步提高效率而使用 P2:

$\text{max}(X, Y, Y) :- X \leq Y, !.$

$\text{max}(X, Y, X).$

虽然过程性语义没有变，但是说明性语义却变成了：“X, Y 的最大值总是 X；如果 $X \leq Y$ ，还可以是 Y”。显然这样的说明性语义已经不是求最大值的原意了。

在 Prolog 中有一个内部谓词 fail，它作为一个目标总是失败并引起回溯。当 fail 前面是 cut 时，则由于 cut 的作用是改变正常的回溯方式，使匹配含该 cut 规则的父目标立即失败，这样提高了搜索效率。

例如，对于下面的程序 P:

(1) $\text{strong}(X) :- \text{heart_disease}(X), \text{fail}.$

(2) $\text{strong}(X) :- \text{tuberculosis}(X), \text{fail}.$

(3) $\text{strong}(X) :- \text{nearsight}(X), \text{fail}.$

(4) $\text{strong}(X).$

(5) $\text{heart_disease}(\text{Zhang}).$

如询问: $?- \text{strong}(\text{Zhang})$ ，第一句中 heart_disease 谓词匹配成功。但是后面的 fail 总是失败，引起回溯。这样就造成 Prolog 去匹配后面的子句，最后因为第四条匹配成功而结束。无疑这是 Prolog 的回溯功能引起的结果，为了停止 Prolog 这样不必要的回溯，可以用 CUT 放在 fail 之前:

(1) ' $\text{strong}(X) :- \text{heart_disease}(X), !, \text{fail}.$

(2) ' $\text{strong}(X) :- \text{tuberculosis}(X), !, \text{fail}.$

(3) ' $\text{strong}(X) :- \text{nearsight}(X), !, \text{fail}.$

则上述目标匹配第一条规则后，因为 fail 马上失败，又因为 cut 使得从父目标 strong(Zhang)到 cut 之间所有目标都是不可重新满足的，因此系统回答为 NO。这就是 cut-fail 的组合。

一阶逻辑的不可判定性告诉我们，对于程序 P 和目标 G ，不存在算法，在有限步内报告 G 是不是 P 的逻辑推论。当然如果 G 是 P 的逻辑结论，SLD 归结法会在有限步内告诉我们，但若 G 不是 P 的逻辑推论，SLD 法（或其他方法）可能会陷入死循环。为此，在逻辑程序设计中引入了一条规则：非即失败。该规则即是说：对于子句 G ，如果无法证明 G ，则推出 $\neg G$ 是合理的。

在 Prolog 中定义了谓词 `not`，`not(A)` 即 $\neg A$ ，它可定义为：

```
not(A) :- call(A), !, fail.  
not(A).
```

`call` 是一个内部谓词，`call(A)` 要求系统把 A 作为一个子目标求解。上面的定义是说：如果 A 有解，则 `not(A)` 无解，若 A 无解，则 `not(A)` 成功。显然该定义是非即失败的正确描述。

2.3 非单调逻辑

20 世纪 80 年代以来，智能科学的需求促使各种非经典逻辑系统的出现和发展，非单调逻辑就是其中的典型例子 [McDermott 1980]。

人类认识世界的过程是个否定之否定的辩证发展过程。由于人类对客观世界的认识不仅是不确定的，而且往往是不完全的，尽管人的知识就总的认识过程来看是在不断增长着的，但新知识的出现往往在一定程度上否定、完善、补充旧的知识。正如波普尔所指出的那样，科学的发现过程是个证伪过程，在一定的条件和环境下，一个理论的形成总存在着它的历史局限性。随着人类认识世界的逐渐深入，随着科学研究的不断发展，旧的理论不能满足新的需要，往往会被新的发现所证伪；于是旧的理论被否定，新的理论就诞生。从这个意义上来说，人类知识的增长过程实际上是非单调的发展过程。

然而，经典逻辑，如形式逻辑、演绎逻辑等，对人类认识世界的处理却是单调的。也就是说，在现有知识的基础上，通过严密的逻辑论证和推理获得的新知识必须与已有的知识相一致。具体地说，设有知识系统 A ，如果已知 A 蕴涵着知识 B ，即 $A \rightarrow B$ ，则可推理得出知识 B 。在此过程中，严格要求 B 必须遵从知识系统 A 。然而正如上面所述，人类的知识并不严格遵循上述约定，因为它非单调的。

所谓非单调推理就是具备这样一种特性的推理：推理系统的定理集并不随推理过程的进行而单调地增大，新推出的定理很可能会否定、改变原来的一些定理，使得原来能够解释的某些现象变得不能解释了。假如把人们在不同认识阶段的知识用集合 F 来表示，则这样的集合是时间 t 的函数 $F(t)$ 。每个集合 $F(t)$ 表示人们在时刻 t 的知识总和，则这些集合不是单调增大的。形式地说，如果 $t_1 < t_2$ ，则 $F(t_1) \subseteq F(t_2)$ 并不成立。然而人们的知识却一直在不断增长。导致这一现象的根本原因就是人们推理时所依据的知识具有不完全性。非单调逻辑是处理不完全知识的工具。

单调逻辑的推理规则是单调的。设 Γ 表示推理规则集，则单调逻辑的语言 $Th(\Gamma) = \{A \mid \Gamma \rightarrow A\}$ 具有如下单调性：

- (1) $\Gamma \in \text{Th}(\Gamma)$
- (2) 如果 $\Gamma_1 \subseteq \Gamma_2$, 则 $\text{Th}(\Gamma_1) \subseteq \text{Th}(\Gamma_2)$
- (3) $\text{Th}(\text{Th}(\Gamma)) = \text{Th}(\Gamma)$ (幂等性)

其中 (3) 又称为不动点 (fixed point)。单调推理规则的显著特性之一就是它的语言是封闭的最小不动点, 亦即 $\text{Th}(\Gamma_1) = \bigcap \{S \mid \Gamma_1 \rightarrow S \text{ 且 } \text{Th}(S) = \Gamma_2\}$ 。

为了处理非单调性, 需要一个新的推理规则, 其形式为:

- (4) 如果 $\Gamma \vdash \neg P$, 则 $\Gamma \vdash MP$

其中 M 是个模态词, 其意义为: 如果在 Γ 中不能证明 $\neg P$, 则默认为 P 在 Γ 中为真。

显然使用单调推理系统无法保证: 当 Γ 含有推理规则 (4) 时仍具有固定点 $\text{Th}(\Gamma) = \Gamma$ 。为此, 我们定义操作符 NM 如下: 对任意一阶理论 Γ 和公式集 $S \subseteq L$, 令:

- (5) $NM_{\Gamma}(S) = \text{Th}(\Gamma \cup A_{S\Gamma}(S))$

其中 $A_{S\Gamma}(S)$ 为 S 的假设集, 由下式给出:

- (6) $A_{S\Gamma}(S) = \{MP \mid P \in L \wedge P \in S\} - \text{Th}(\Gamma)$

于是可定义 $\text{Th}(\Gamma)$ 为从 Γ 非单调地推出的定理集合如下:

- (7) $\text{Th}(\Gamma) = NM_{\Gamma}$ 的最小固定点

引入式 (7) 的目的在于试图将推理规则 (4) 加入到一阶理论 Γ 之中, 使之能进行封闭的推理。但实际上由于 Th 的定义要求太强, 而 $\text{Th}(\Gamma)$ 不仅无法求得, 而且也无法保证 $\text{Th}(\Gamma)$ 一定存在, 为此将式 (7) 修改为:

- (8) $\text{Th}(\Gamma) = \bigcap (\{L\} \cup \{S \mid NM_{\Gamma}(S) = S\})$

这样, 如果 L 是该逻辑的语言, 则 L 恒为固定点, 因为 $NM_{\Gamma}(L) = L$ 恒成立, 通常称为平凡固定点。因此当不存在有非平凡固定点时 $\text{Th}(\Gamma) = \bigcap (\{L\}) = L$ 。

实际上我们可以证明如果 $\text{Th}(\Gamma)$ 不存在, 则说明 Γ 中含有矛盾。下面为了处理和说明上的方便, 我们把式 (8) 等价地写成:

- (9) $\text{Th}(\Gamma) = \{P \mid \Gamma \vdash P\}$

其中 $\Gamma \vdash P$ 表示 $P \in \text{Th}(\Gamma)$, 并把 $\{S \mid NM_{\Gamma}(S) = S\}$ 简写成 $FP(\Gamma)$, 称该集合中的元素为理论 Γ 的固定点。

非单调推理有三个主要流派, 即 McCarthy 提出的限制理论: 当且仅当没有事实证明 S 在更大的范围成立时, S 只在指定的范围成立; Reiter 的缺省逻辑: “ S 在缺省的条件下成立”是指“当且仅当没有事实证明 S 不成立时 S 是成立的”。Moore 的自认知逻辑: “如果我知道 S , 并且我不知道有其他任何事实与 S 矛盾, 则 S 是成立的”。

对逻辑进行扩展, 将非单调推理形式化, 称为非单调逻辑。它包括语言方面的扩充和语义方面的扩充, 语言方面的扩充是指增强其表达能力, 语义方面的扩充是指对真值的真假两种情况进行修正; 一是对推理模式的扩展, 这涉及非单调推理的过程化方面, 称为非单调系统。

非单调逻辑大致分为两类: 一类基于最小化语义, 称为最小化非单调逻辑; 另一类基于定点定义, 称为定点非单调逻辑。

最小化非单调逻辑可以分为基于最小化模型和基于最小化知识模型。前者主要有封闭世

界假设、McCarthy 的限制逻辑 (circumscription) 等, 后者包括 Konolige 的忽略逻辑 (ignorance) 等。

定点非单调逻辑可以分为缺省逻辑 (default) 和自认知逻辑 (autoepistemic)。McDermott 和 Doyle 提出的非单调模态逻辑 NML 旨在研究非单调逻辑的一般基础, 是一种一般缺省逻辑。Reiter 的缺省逻辑则是对缺省规则的一阶形式化。自认知逻辑是 Moore 提出的, 是对 McDermott 和 Doyle 的非单调逻辑语义困难的一种改进。

非单调系统的实现, 可以通过对矛盾的检测进行真值的修正来维护相容性, 可称为真值维护系统, 包括 Doyle 提出的真值维护系统 TMS, Dekleer 提出的基于假设的真值维护系统 ATMS 等等。

2.4 封闭世界假设

封闭世界假设 (Closed World Assumption—CWA) 是一种对由一组基本信念集合 KB 定义的理论 $T(KB)$ 进行完备化的方法。一个理论 $T(KB)$ 是完备的, 是说其包含 (显式或隐含) 了每一个基原子公式或该公式否定。CWA 的基本思想是: 如果无法证明 P, 则认为它是否定的。即:

如果从知识库中无法证明 P 或者 $\neg P$, 则就向 KB 中增加 $\neg P$ 。

这就是说你假定知道所有有关世界的事情 (即世界是封闭的)。

CWA 的最大用处是完备化数据库系统。例如, 我们可以设计一个关于国家邻接的数据库 Neighbor(x, y)。基于 CWA, 凡是未在该数据库中说明是邻接的国家都是不邻接的。

我们知道理论 $T(KB)$ 是一组在逻辑蕴涵下封闭的句子的集合 (如传递封闭)。理论 $T(KB)$ 是完备的, 如果语言中每个基原子本身或者是其否定在该理论中。

假定 KB: Neighbor(China, Russia).
 Neighbor(China, Mongolia).
 $\forall x \forall y (\text{Neighbor}(x, y) \leftrightarrow \text{Neighbor}(y, x))$

则 $T(KB)$ 是不完全的, 因为无论是 Neighbor(Russia, Mongolia) 还是 $\neg \text{Neighbor}(\text{Russia, Mongolia})$ 都不在 KB 中。

CWA 对理论的完备化是仅仅通过向基本信念集合 KB 中增加基原子公式的取反来实现的。换言之, 若一个基原子公式不能经由逻辑推理从基本信念集 KB 导出, 就将其取反作为 KB 的扩充。显然, CWA 是非单调的, 因为一旦以后有新的基原子公式加进 KB, 则为完备 $T(KB)$ 而生成的扩充集就必须收缩 (删除该基原子公式的否定)。例如对于国家相邻问题, 可以向 KB 中增加 $\neg \text{Neighbor}(\text{Russia, Mongolia})$ 实现完备化。

由于为完备 $T(KB)$ 而生成的扩充集中的每个基原子公式的取反均是假设的暂时信念, 故记该扩充集为 KB_{asm} 。对于一个基原子公式 P (省略其常量项), CWA 定义:

$\neg P \in KB_{asm}$, 当且仅当 $P \notin T(KB)$ 。

记经由 CWA 方法完备的理论为 $CWA(KB)$, 其扩大了 $T(KB)$ 的推理能力, 允许不能从 KB 导出的结论 ϕ 可从 $KB \cup KB_{asm}$ 导出。在上例中, $KB_{asm} = \{\neg \text{neighbor}(\text{Russia, Mongolia})\}$, $CWA(KB) = T(KB \cup KB_{asm})$ 。

然而 CWA 方法并不确保被完备的理论 $CWA(KB)$ 是一致的。例如, 令 $KB = \{P(A) \vee P(B)\}$, 因为无法推出 P(A) 或者 $\neg P(A)$, 以及 P(B) 或者 $\neg P(B)$, 所以有 $KB_{asm} = \{\neg P(A), \neg P(B)\}$ 。但是 $KB \cup KB_{asm}$ 是不一致的, 它会导致 $P(A) \vee P(B)$ 为假。解决不一致性是非单调推理的重要议题, 需要对 CWA 的完备性规则进行修改, 以实现一致性。

向一个一致的 KB 中增加新的句子, 为了保持增加句子后 KB 的一致性, 可以通过下述定

理试图解决不一致性。

定理 2.1 $CWA(KB)$ 是一致的，当且仅当对于每个可由 KB 推导出的子句 $P_1 \vee P_2 \vee \dots \vee P_n$ ，都至少存在一个 P_i 可从 KB 推导出；其中 P_i 均为正基文字。

也就是说， $CWA(KB)$ 是不一致的，当且仅当存在正基文字 P_1, P_2, \dots, P_n ，有 $KB \models P_1 \vee P_2 \vee \dots \vee P_n$ ，但是对所有的 i ， $KB \not\models P_i$ 。

通过把所有的基文字增加到 KB 中以实现完备性，并不和该定理矛盾。

例 2.2 $KB: P(A) \vee P(B)$ ， $CWA(KB)$ 是不一致的

例 2.3 $KB: \forall x(P(x) \vee Q(x))$

$P(A)$

$Q(B)$

对原子 A 和 B ，扩充 KB 包括： $\neg P(B)$ 和 $\neg Q(A)$ 。对于原子 C ，这样的扩充就是不一致的，因为 $(P(x) \vee Q(x)) \not\models P(C)$ 并且 $(P(x) \vee Q(x)) \not\models Q(C)$ 。

总的来说，由 CWA 扩充的知识库也可能是 inconsistent 的。然而，如果知识库是由 Horn 子句组成，并且是一致的，则经过封闭世界扩充就是一致的。即：

定理 2.2 若 KB 是由一致的 Horn 形子句组成时，则 $CWA(KB)$ 必定一致。

在第四章，我们知道 Horn 子句定义为文字 P_i 的析取形式： $P_1 \vee P_2 \vee \dots \vee P_n$ ，且最多只能有一个 P_i 是正文字。然而，这对于许多应用来讲，限制太强，如果实用上仅对某一特殊谓词 P 感兴趣，则可减弱定理 2.2 的强度，只将谓词为 P 的基原子公式的取反加入 KB_{asm} ，而 Horn 子句也减弱到仅对谓词为 P 的文字满足 Horn 子句定义。

例如， $KB = \{P(A) \vee Q(A), P(A) \vee R(A)\}$ ， KB 包含的两个子句均非 Horn 子句，但都是关于谓词 P 的广义 Horn 子句；令 $KB_{asm} = \{\neg P(A)\}$ ，则可得到关于谓词 P 的扩充理论 $CWA'(KB)$ ，并有 $KB \cup KB_{asm} \models Q(A), KB \cup KB_{asm} \models R(A)$ 。

这样可以得到关于一个谓词集 I 的 $CWA'(KB)$ ，此时 KB 包含的每个子句均为关于 I 的广义 Horn 子句，即每个子句中最多只能出现一个其谓词属于 I 的正文字。问题是，关于谓词集 I 的 $CWA'(KB)$ 仍不能确保一致性。例如，令 $I = \{P\}$ ， $KB = \{P(A) \vee Q, P(B) \vee \neg Q\}$ ；则有 $KB_{asm} = \{\neg P(A), \neg P(B)\}$ 。由于 $KB \models P(A) \vee P(B)$ ，故 $CWA'(KB)$ 是不一致的。

2.5 默认逻辑

默认推理是一类似然推理。从直观上理解，各种默认推理都企图对以下形式的断言进行推理：

正常情况下 A 成立，

典型情况下 A 成立，

默认假定 A 成立。

默认推理的一个典型例子是“鸟会飞”。“鸟会飞”这个陈述并不等于“所有鸟都会飞”这个陈述，因为有许多反例，例如鸵鸟、企鹅等等。对于某个具体的鸟，例如说 a ，我们可能会根据似然命题：

正常情况下 鸟会飞, 或

典型情况下 鸟会飞, 或

如果 x 是鸟, 那么默认假定 x 会飞

来推断 a 会飞。但如果随后我们又获得相反的信息, 比如说 a 是一个鸵鸟, 我们会撤消前面的结论, 并作出结论 a 不会飞。而在开始我们是假定 a 会飞。显然, 这里所体现的是似然推理, 而不是演绎推理。

1977 年加拿大多伦多大学的 Reiter 研究不完全信息的推理形式, 并于 1980 年正式提出了默认逻辑 (default logic) [Reiter 1980]。

在非单调推理中, 缺省推理 (也称为默认逻辑) 是应用比较广泛的一种逻辑, 它是在信息不完全和前提缺省的情况下, 默认一些先决条件而进行的推理。在计算机程序设计中, 缺省是经常使用的一种技术手段, 其目的是为了给程序员提供方便, 如在程序设计中, 内、外层程序出现同一标示符 x , 则内层标示符 x 的缺省类型就是外层 x 的类型, 除非 x 在内层被重新说明。这时缺省的含义是: 如果程序员不显式地指明某种要求, 则系统将按约定的章程办事。

缺省是表达非单调强有力的手段, 为此 Reiter 把缺省概念引入到逻辑中设计了缺省逻辑, 其基本思想是: 传统的逻辑是从已知的事实推出新的事实, 在推理时, 知识库的丰富程度决定了能推出多少事实。而在非单调推理中, 知识库不够丰富, 难以支持系统所需要的推理, 因此需要对知识库进行扩充, 这些扩充的知识就是缺省的知识。

和 CWA 类似, 这些缺省的知识并不十分可靠, 只是在目前看来不和知识库的其他部分发生矛盾, 所以推出来的不能算是事实, 只是对现实世界的一种猜测。Reiter 把原来的知识库称为不完备的理论, 把扩充看作是理论的完备化。而缺省逻辑则提供了一组元公理, 作为理论完备化的手段。Reiter 为此做了两方面的研究: 首先对不完备理论的扩充给出了形式化的定义; 其次是探讨了缺省逻辑的证明论问题, 即在什么条件下可以推断一个命题是否属于某个不完备理论的一个扩充。

Reiter 的缺省逻辑是另一种形式的非单调逻辑, 它包括一阶的语句以及一个或多个缺省假设。形式地说, 缺省逻辑 DL 是对基本的知识库 KB 扩充一组非标准的、非单调的推理规则 D 实现的。扩充了 D 的 KB 包括标准逻辑所具有的结论加上应用 D 于 KB 所得到的结论。

DL 系统 D 中的缺省规则形式为:

$$\frac{\alpha(\bar{x}) : M\beta_1(\bar{x}), \dots, M\beta_m(\bar{x})}{W(\bar{x})} \quad (2.1)$$

或者表示为线性形式为:

$$\alpha(\bar{x}) : M\beta_1(\bar{x}), \dots, M\beta_m(\bar{x}) \rightarrow W(\bar{x}) \quad (2.2)$$

其中 \bar{x} 是由 x_i 构成的参数向量。 $\alpha(\bar{x})$ 是命题的前提, $W(\bar{x})$ 是命题的结论, $\beta_i(\bar{x})$ 是缺省条件, M 为缺省算子, 表示相容性, 即前提和缺省条件相容或不矛盾, 就可以推出结论。所以缺省规则 D 可以读为: 如果没有信息表明 $\beta_1(\bar{x}), \dots, \beta_m(\bar{x})$ 中任何一项不成立 (或与现有的知识相矛盾), 则可从前提 $\alpha(\bar{x})$ 推出结论 $W(\bar{x})$ 。例如对于缺省规则:

$$\frac{bird(x) : M flies(x)}{flies(x)}$$

它表示：如果 x 是鸟，并且如果认为 x 会飞不会引起矛盾，则可以相信 x 就会飞。缺省规则在表达通常情况下是正确的，但并非绝对是正确的信念时非常有用。如果缺省规则中不含自由变元，即 α 、 $M\beta_i$ 、 W 都是命题，那么该规则称为闭规则。

定义 2.7 一个缺省理论 T 由两个部分组成：

- 1、缺省推理规则集合 D
- 2、公式集合 W ，它是已知的或约定的事实的集合

当 D 中所有规则都是闭规则时，称理论 $T=\langle D, W \rangle$ 为闭缺省理论。

对于缺省理论 $T=\langle W, D \rangle$ ，假设 $D = \frac{MA}{B}$ ， $W=\Phi$ ，那么 B 在 T 中可以推出。向 W 增加 $\neg A$ ，则 W 变为 $W'=\{\neg A\}$ ， $T'=\langle D, W' \rangle$ 。这是尽管 T' 是 T 的扩充（已知事实集合 $W' \supseteq W$ ），但是 B 却不能从 T' 中推出。所以缺省理论 T 是非单调的。

例 2.4 假设 W 为：
 $bird(tweety)$
 $\forall x(ostrich(x) \rightarrow \neg flies(x))$
 D 为：
$$\frac{bird(x) : M \quad flies(x)}{flies(x)}$$

则可以得到 $flies(tweety)$ 。

如果在 W 中增加 $ostrich(tweety)$ ，则无法得到 $flies(tweety)$ 。

例 2.5 假设 W 为：
 $feathers(tweety)$

D 为：
$$\frac{bird(x) : M \quad flies(x)}{flies(x)}, \frac{feathers(x) : M \quad bird(x)}{bird(x)}$$

则同样可以得到 $flies(tweety)$ 。

如果在 W 中增加 $ostrich(tweety)$ ，

$\forall x(ostrich(x) \rightarrow \neg flies(x))$
 $\forall x(ostrich(x) \rightarrow feathers(x))$

则也无法得到 $flies(tweety)$ 。

在缺省理论中，“推出”概念和传统逻辑中的“推出”概念是有区别的，前者是非单调推理而后者是单调推理。为了定义缺省理论中的“推出”概念，需要下面的定义。

定义 2.8 设 $\Delta=\langle D, W \rangle$ 为一闭缺省理论， Γ 为关于 D 的一个算子， Γ 作用于任意的命题集合 S ，其值为满足下面三个性质的最小命题集合 $\Gamma(S)$ ：

- (1) $W \subseteq \Gamma(S)$;
- (2) $\Gamma(S)$ 为在普通命题演算的推理下封闭的，即 $Th(\Gamma(S)) = \Gamma(S)$;
- (3) 如果 D 中有规则： $\alpha : M\beta_1, \dots, M\beta_m \rightarrow w$ ，且 $\alpha \in \Gamma(S)$ ， $\neg\beta_1, \dots, \neg\beta_m \notin S$ ，那么 $w \in \Gamma(S)$ 。

定义 2.9 命题集合 E 称为关于 D 的算子 Γ 的固定点，如果 $\Gamma(E)=E$ 。此时又称 E 为 $\Delta=\langle D, W \rangle$ 的一个扩张。

有了扩张的概念，便可以定义非单调推理中的“推出”概念。

定义 2.10 如果命题 A 包含在缺省理论 $\Delta=\langle D, W \rangle$ 的一个扩张中，那么称 A 在 Δ 中可以推出，记为 \sim ，表示非单调“推出”。（可见例 5.1 中的关系 R ）

例 2.6 设 $D = \left\{ \frac{MA}{\neg A} \right\}$, $W = \Phi$ 。则 $\Delta = \langle D, W \rangle$ 无扩张。

因为可以证明关于 Δ 的算子 Γ 无固定点。否则, 假设 E 为 Γ 的固定点。

如果 $\neg A \notin E$, 则由定义 5.2 的 (3) 可以得到 $\neg A \in E$ 。

如果 $\neg A \in E$, 因为 $W = \Phi$, 则 $\neg A$ 一定是由缺省规则 D 导入到 E 的。所以, $\neg A \notin E$, 否则, MA 为假, 该规则不可使用。

显然这是一个悖论。

例 2.7 设 $D = \left\{ \frac{MA}{\neg B}, \frac{MB}{\neg C}, \frac{MC}{\neg F} \right\}$, $W = \Phi$ 。则 $\Delta = \langle D, W \rangle$ 有唯一的扩张 $E = \text{Th}(\{\neg B, \neg F\})$ 。

容易验证 E 为关于 Δ 的 Γ 固定点, 而当命题集 $S \in \{\neg B, \neg C, \neg F\}$ 而 $S \neq \{\neg B, \neg F\}$ 时, $\text{Th}(S)$ 都不是 Γ 的关于 Δ 的固定点。其直观解释为: 第一个缺省规则的结果阻止了第二个缺省规则的应用, 而这有使得第三个缺省规则的应用成为可能。

例 2.8 设 $D = \left\{ \frac{MA}{A}, \frac{B:MC}{C}, \frac{F \wedge A:ME}{E}, \frac{C \wedge E:M\neg A, M(F \vee A)}{G} \right\}$,

$W = \{B, C \rightarrow F \vee A, A \wedge C \rightarrow \neg E\}$ 。

则 $\Delta = \langle D, W \rangle$ 有三个扩张:

$E_1 = \text{Th}(W \cup \{A, C\})$

$E_2 = \text{Th}(W \cup \{A, E\})$,

$E_3 = \text{Th}(W \cup \{C, E, G\})$

仅对 E_1 进行说明。由于 $\text{Th}(W \cup \{A, C\})$ 中有 $\neg E$, 从而 $\Gamma(\text{Th}(W \cup \{A, C\}))$ 中没有 E 和 G (因为 ME 和 $C \wedge E$ 不能成立)。这就是说 D 中的缺省规则在算子 Γ 的计算过程中无一可用, 所以 $\Gamma(\text{Th}(W \cup \{A, C\})) = \text{Th}(W \cup \{A, C\})$ 。

从上面可以看到, 并非所有的缺省理论都有扩张。有扩张的缺省理论也不一定是只有唯一的扩张, 因为该理论中可能有多个缺省规则, 而这些缺省规则的合理条件是不相容的。一个理论是否具有扩张, 决定着在该理论上能否进行有效的缺省推理。因此研究和探讨扩张的存在条件是十分重要的。

定理 2.3 设 E 为一阶命题集合, $\Delta = \langle D, W \rangle$ 为一闭缺省理论。递归定义 $E_i (i=0, 1, 2, \dots)$ 如下:

$E_0 = W$

$E_{i+1} = \text{Th}(E_i) \cup \{w \mid (\alpha: M\beta_1, \dots, M\beta_m \rightarrow w) \in D, \\ \alpha \in E_i, \neg\beta_1, \dots, \neg\beta_m \notin E_i\}$

那么 E 是 Δ 的一个扩张当且仅当 $E = \bigcup_{i=0}^{\infty} E_i$

可以利用该定理来验证例 5.8 中的三个扩张。

现在如果缺省规则为: $\frac{M\neg A}{\neg A}$, 缺省理论是否和 CWA 一样呢? 答案是否定的。

假设 $W = \{P \vee Q\}$, D 为: $\frac{M\neg P}{\neg P}$ 以及 $\frac{M\neg Q}{\neg Q}$ 。显然 $\text{CWT}(\Delta)$ 是不一致的。而 Δ 的扩张可

以是 $\{P \vee Q, \neg P\}$ 或者为 $\{P \vee Q, \neg Q\}$, 但如果把它们合在一起就不一致了。

例 2.9 设 $D = \left\{ \frac{MA}{\neg A} \right\}$, $W = \{A, \neg A\}$ 。则 $\Delta = \langle D, W \rangle$ 的扩张 $E = \text{Th}(W)$, 即维持不变。

该例子与前面的例子不同，它得到的扩张是不一致的（既有 A，又有 $\neg A$ ）。根据数理逻辑的基本知识知道，这个扩张包括了该系统中所有可能的一切合式公式。

关于不一致的扩张有下面一些结论：

- 1、当且仅当 W 本身是不一致的时候，闭缺省理论 $\langle D, W \rangle$ 有一个不一致的扩张；
因为 $W \subseteq E$ ，所以如果 W 不一致，则 E 也不一致，即充分性成立。反之，如果 E 不一致，则 E 必包含一切命题，因此所有的缺省规则都可以使用，因此由定理 5.3 得 $E = Th(W)$ ，所以 W 是不一致的。
- 2、若一个闭缺省理论有一个不一致的扩张，则这是它唯一的扩张；

缺省理论的扩张一般是不唯一的，那么它们之间有什么关系呢？

- 3、设 E 和 F 是同一闭缺省理论的两个不同的扩张，则如果 $E \subseteq F$ ，则 $E = F$ 。并且
- 4、如果 $\Delta_1 = \langle D_1, W_1 \rangle$ ， $\Delta_2 = \langle D_2, W_2 \rangle$ 为两个缺省理论，并且 $W_1 \subseteq W_2$ ，如果 Δ_2 的扩张都是一致的，那么 Δ_1 的扩张也是一致的。

这是显然的。若 Δ_1 有不一致的扩张，这 Δ_1 是不一致的（见性质 1），从而 Δ_2 是不一致的（因为 $W_1 \subseteq W_2$ ），它的扩张也是不一致的。

定义 2.11 一个缺省理论 $\Delta = \langle D, W \rangle$ 称为规范，如果 D 中的缺省规则均为如下形式：

$$\frac{A : MB}{B} \quad (2.3)$$

这样的缺省规则称为规范缺省规则。

如果 W 是一致的，由于每个规范缺省规则的结论和缺省条件相同，所以不会导致不一致性，并且 Reiter 给出下面的结果：

- (1) 任何规范缺省理论必定至少有一个扩张。
- (2) 设 E 和 F 是同一规范缺省理论的两个不同的扩张，则 $E \cup F$ 是不一致的。
这说明，如果推理者对他所在的世界有两种不同的想象，那么它们一定是不一致的。
- (3) 设 $\Delta = \langle D, W \rangle$ 为闭规范缺省理论， $D' \subseteq D$ ，且 E'_1 、 E'_2 分别为 $\langle D', W \rangle$ 的两个不同的扩张，那么 Δ 必有不同的扩张 E_1 、 E_2 ，使得 $E'_1 \subseteq E_1$ 、 $E'_2 \subseteq E_2$ 。
这说明闭缺省规范理论的扩张的大小，随闭缺省规则数目的增加而单调不减。

如果缺省规则形式为：

$$\frac{A : MB \wedge MC}{B} \quad (2.4)$$

则称为半规范缺省规则。如果一个理论所有的缺省规则要么是规范的，要么是半规范的，且至少有一个是半规范的，则该理论称为半规范缺省理论。

对于有序缺省理论 (ordered default theory)，需要考虑偏序关系：优先关系 “ \ll ” 和 “ $\ll=$ ”，其中前者称为强优先关系，后者称为弱优先关系：如果在推导 B 的过程中，用到了 C，那么有： $C \ll B$ 。

例如： $D = \left\{ \frac{A : M(B \wedge \neg C)}{B} \right\}$ ，B 是强优先于 C，记为 $C \ll B$ 。

一个半缺省理论是有序的，如果不存在文字 Y，使得 $Y \ll Y$ 。例如：

$$D = \left\{ \frac{A : M(A \wedge \neg B)}{A}, \frac{B : M(B \wedge \neg C)}{B}, \frac{M(C \wedge \neg A)}{C} \right\}$$

则有 $B \ll A$ ， $C \ll B$ ， $A \ll C$ ，结果有 $A \ll A$ ，因此该缺省理论不是有序的。

半规范缺省理论至少具有一个扩张的充分条件是要求闭半规范缺省理论是有序的。

2.6 限制逻辑

限制逻辑(CIRC, circumscription logic)是 McCarthy 提出的一种非单调逻辑。限制的基本思想是“从某些事实 A 出发能够推出具有某一性质 P 的对象就是满足性质 P 的全部对象”[McCarthy 1980]。在常识推理中,人们经常把已发现的、具有某些性质的对象,看作具有该性质的全部对象,并据此进行推理。只有当发现其它对象也具有该性质时,才修改这种看法。这是一种非单调的推理形式,在常识推理中占有极为重要的地位。例如,大数学家 Erdos 曾猜想不定方程 $x^x y^y = z^z$ 只有平凡解 $x=1, y=z$ 和 $y=1, x=z$ 。后来,我国数学家柯召给出了该方程的无穷多个非平凡解,推翻了 Erdos 的猜想。Erdos 又提出新的猜想,找到该方程的全部解。

限制逻辑 CIRC 是一种极小化逻辑。下面,从一个基于极小模型定义的命题限制出发,给出限制的基本定义,进而给出一阶限制的基本结果,并将它推广。

定义 2.12 设 L_0 是一个命题语言, p_1, p_2 是在命题语言 L_0 中的两个赋值。称 p_1 小于 p_2 , 记为 $p_1 \preceq p_2$, 当且仅当对任一命题变元 x , 如果 $p_1(x) = 1$, 则 $p_2(x) = 1$ 。

定义 2.13 设 A 是一个公式, 称 A 的一个赋值 p 是极小的, 当且仅当不存在 A 的其它赋值 p' 使得 $p' \preceq p$ 。

显然, \preceq 是一个偏序关系。 $p_1 \preceq p_2$ 表示 p_1 包含的真命题比 p_2 少。极小赋值包含的真命题极小。

定义 2.14 极小后承 \vdash_M 。设 A, B 是两个公式, $a \vdash_M B$ 当且仅当 B 在所有 A 的极小模型中都为真。

极小模型是非单调的, 它以命题的极小化作为优先模型的准则。它的性质可由下面简单的例子看出。例如

$$p \vdash_M \neg q$$

$$p \vee q \vdash_M \neg p \vee \neg q$$

$$p, q, p \vee q \vdash_M p \wedge q$$

命题限制可定义为对于一个命题集 Z 的动态极小, 而使不在 S 中的命题变元动态可变。

定义 2.15 设 A 是一个包含命题集 $P = \{p_1, p_2, \dots, p_n\}$ 的公式, 一个 A 的赋值 p 称为 \preceq^Z -极小赋值, 当且仅当不存在 A 的其它赋值 p' 使得 $p \preceq p'$, 定义如下: 设 p_1, p_2 是两个赋值, $p_1 \preceq^Z p_2$ 当且仅当对任一 $z \in Z$, 若 $p_1(z) = 1$, 则 $p_2(z) = 1$ 。

定义 2.16 命题限制 \vdash_P 或 $\text{CIRC}(A, P)$ 。设 A 是一个包含命题集 P 的公式, ϕ 是一个公式, a

$\vdash_P \varphi$ 当且仅当 φ 在所有 A 的 \succeq^P -极小赋值中都为真。

命题限制 $\text{CIRC}(A, P)$ 可用如下一阶句子描述:

$$A(P) \wedge \forall P' (A(P') \wedge P' \rightarrow P) \rightarrow (P \rightarrow P') \quad (2.5)$$

其中, P' 是与 A 协调的 P 等替物, $\forall P'$ 是一个命题全称量词, $a(P')$ 是在 A 中以 P' 替代 P 的结果。如果用 $P' \succ_P P$ 表示 $P' \rightarrow P$, 则 $\text{CIRC}(A, P)$ 可以写为:

$$A(P) \wedge \neg \exists P' (A(P') \wedge P' \succ_P P) \quad (2.6)$$

命题限制中一个逻辑推论记为 $A \vdash_P \varphi$, 或者 $\text{CIRC}(A, P) \vdash_P \varphi$ 。具有如下的可靠和完全性定理。

定理 2.4 $A \vdash_P \varphi$ 当且仅当 $A \vdash \varphi$ 。

命题限制太弱。在此基础上可以推广到一阶限制。一阶限制是基于最小模型的思想。

定义 2.17 令 L 是一个一阶语言, T 是一个 L 的公式, 它包含谓词元组集 ρ 。设 $M[T]$ 和 $M^*[T]$ 是公式 T 的两个模型。定义 $M^*[T]$ 优先于 $M[T]$, 记为 $M^*[T] \succeq_\rho M[T]$, 当且仅当

- (1) M 和 M^* 有相同的对象域,
- (2) 除 ρ 外, 公式 T 中所有的其它关系和函数常数在 M 和 M^* 都有相同的解释,
- (3) ρ 在 M^* 中的外延是 ρ 在 M 中的子集。

一个理论 T 的模型 M 称为优先的, 当且仅当不存在 T 的其它模型 M' 使得 $M' \succeq_\rho M$ 。

定义 2.18 M_ρ 是 ρ 的最小模型, 当且仅当 $M \succeq_\rho M_\rho$, $M = M_\rho$ 。

例如, 设论域 $D = \{1, 2\}$ 上,

$$\begin{aligned} T &= \forall x \exists y (P(y) \wedge Q(x, y)) \\ &= [(P(1) \wedge Q(1, 1)) \vee (P(2) \wedge Q(1, 2))] \wedge \\ &\quad [(P(1) \wedge Q(2, 1)) \vee (P(2) \wedge Q(2, 2))] \end{aligned}$$

若 M :	$P(1)$	$P(2)$	$Q(1, 1)$	$Q(1, 2)$	$Q(2, 1)$	$Q(2, 2)$
	True	True	False	True	False	True
M^* :	$P(1)$	$P(2)$	$Q(1, 1)$	$Q(1, 2)$	$Q(2, 1)$	$Q(2, 2)$
	False	True	False	True	False	True

由此可知, 模型 M 和 M^* 对谓词 Q 的真值设定是相同的, 而对 P 来说, M 中有 $\{1, 2\}$ 中的元素使 P 为真, 而 M^* 中仅有 $\{2\}$ 中的 2 使 P 为真, $\{2\} \subseteq \{1, 2\}$ 。于是有 $M^* \succeq_\rho M$ 。如果 $M^* \succeq_\rho M$, 而且 $M^* \neq M$, 则有 $M^* \succ_\rho M$ 。对理论 T 来说, 如果对任一 $M \succeq_\rho M_\rho$, 必有 $M = M_\rho$ 时, 就说 M_ρ

是 P 的最小模型，对任一个 T 来说，最小模型不一定总存在。

设含有谓词 P 的信任集 T ，寻找对 T 的假设(扩充)公式 φ_P ，使得对 $T \wedge \varphi_P$ 的任一模型 M ，不存在 T 的模型 M^* 满足

$$M^* \succ_P M$$

或说扩充后 $T \wedge \varphi_P$ 的模型对 P 来说不能比原来 T 的模型来得大，是一种最小的扩充。依这最小化原则所得的 $T \wedge \varphi_P$ 便是 T 对 P 的限制。

设 P^* 是某个谓词常项，它与 P 有同样的变元个数，由 P^* 来构造 φ_P ，可指出，

$$(\forall x P^*(x) \rightarrow P(x)) \wedge \neg(\forall x P(x) \rightarrow P^*(x)) \wedge T(P^*)$$

的任一模型都不是 T 对 P 的最小模型。从而

$$\neg((\forall x P^*(x) \rightarrow P(x)) \wedge \neg(\forall x P(x) \rightarrow P^*(x)) \wedge T(P^*))$$

的任一模型是 T 对 P 的最小模型。于是

$$\varphi_P = \forall P^* \neg((\forall x P^*(x) \rightarrow P(x)) \wedge \neg(\forall x P(x) \rightarrow P^*(x)) \wedge T(P^*))$$

为 T 对 P 的限制公式。

定义 2.19 一阶限制的语义后承，记为 $T \vdash_{\text{CIRC}} \varphi$ 或 $\text{CIRC}(T, P) \vdash \varphi$ ，当且仅当 φ 在所有 T 的对于 \succeq^P 的最小模型中都为真。理论 T 的限制公式的合取式给出在 T 中的限制 P ：

$$\begin{aligned} \text{CIRC}(T, P) = T \wedge \forall P^* \neg((\forall x) (P^*(x) \rightarrow P(x)) \wedge \neg(\forall x) (P(x) \rightarrow P^*(x)) \wedge T(P^*)) \end{aligned} \quad (2.7)$$

该公式可以变成

$$\begin{aligned} \text{CIRC}(T, P) = T \wedge \forall P^* ((T(P^*) \wedge (\forall x) (P^*(x) \rightarrow P(x))) \rightarrow (\forall x) (P(x) \rightarrow P^*(x))) \end{aligned} \quad (2.8)$$

这个公式是高阶逻辑公式，因为量词 \forall 作用于谓词 P^* ，但在很多情形下可化为一阶逻辑公式。公式

$$\varphi_P = \forall P^* ((T(P^*) \wedge (\forall x) (P^*(x) \rightarrow P(x))) \rightarrow (\forall x) (P(x) \rightarrow P^*(x))) \quad (2.9)$$

是说，如果求得 P^* 使 $T(P^*)$ 成立，又满足 $\forall x (P^*(x) \rightarrow P(x))$ 那么 $\forall x (P(x) \rightarrow P^*(x))$ 便可作为推理的结论。

$\text{CIRC}(T, P)$ 还可写成另一种形式，将 P^* 记作 $P \wedge P'$ (P' 是与 P 的变元个数相同的谓词常

项)，于是有

$$\varphi_P = T(P \wedge P') \vee \forall x (P(x) \wedge P'(x) \rightarrow P(x)) \rightarrow (\forall x) (P(x) \rightarrow P(x) \wedge P'(x)) \quad (2.10)$$

从而得到

$$T(P \wedge P') \rightarrow (\forall x) (P(x) \rightarrow P'(x)) \quad (2.11)$$

若将 $(\forall x) (P^*(x) \rightarrow P(x))$ 记作 $P^* \succeq P$ ，则

$$P^* \succ P \text{ 表示 } (P^* \succeq P) \wedge \neg(P \succeq P^*)$$

$$P^* = P \text{ 表示 } (P^* \succeq P) \wedge (P \succeq P^*)$$

于是 φ_P 就是

$$\varphi_P = \forall P^* (T(P^*) \wedge (P^* \succeq P) \rightarrow (P \succeq P^*)) \quad (2.12)$$

得到

$$\begin{aligned} \varphi_P &= \forall P^* (T(P^*) \rightarrow \neg (P^* \succ P)) \\ &= \neg (\exists P^*) (T(P^*) \wedge (P^* \succ P)) \end{aligned} \quad (2.13)$$

这就更明显地可看出，最小模型限制表明，不存在 P^* 满足 T 而且使 P^* 成立的外延是 P 成立外延的真子集。

定理 2.5 已知谓词 P 和信任集 $T(P)$ ，对任一 P' 来说，如果 $T(P) \vdash T(P') \wedge (P' \succeq P)$ 则

$$\text{CIRC}(T, P) = T(P) \wedge (P = P') \quad (2.14)$$

这定理给出，如果 $T(P)$ 能证明 $T(P') \wedge (P' \succeq P)$ ，那么 $P = P'$ ，就是 T 对 P 的限制公式。

2.7 非单调逻辑 NML

在 1980 年前后，McDermott 和 Doyle 就非单调推理发表了几篇有影响的文章，描述了他们自己所建立的非单调逻辑推理系统，称为 NML[McDermott 1980]。这一系统是建立在一阶逻辑基础上，并引进模态词 \Diamond ，称为相容性操作符。例如：

$$\forall x (\text{Bird}(x) \wedge \Diamond \text{Fly}(x) \rightarrow \text{Fly}(x))$$

上式表示：如果 x 是鸟，并且 x 可以是和现有知识相容的，则 x 会飞。

可以看出 NML 可以处理缺省假设，因此前面介绍的缺省理论可以看作是 NML 的一种特殊情况。但是由于非单调逻辑中允许 $\Diamond A$ 与一般命题一样使用，而缺省理论中 $\Diamond A$ 只能在缺省推理规则中使用，使得 NML 理论与缺省理论有许多根本不同的地方。

下面我们来考虑根据相容性操作符 \Diamond 定义非单调推理机制问题。从语法的角度看，根据 \Diamond 的直观意义，可以有下述规则：

如果 $\neg \neg A$ ，则有 $\neg \Diamond A$

表示如果我们推不出 A 的否定成立，则认为 A 为相容的。但是这样做是不合适的，因为这样

等于把一切非定理的否定接受为定理，没有什么非单调可言了。

McDermott 和 Doyle 修改上式为：

如果 $\vdash \neg A$ ，则有 $\vdash \Diamond A$

这里的 \vdash 符号和缺省理论中是一样的，表示非单调地推出。

我们也可以通过下面解释来说明 \vdash 是否就是一阶谓词的可证明关系 \vdash 。

我们知道，对于单调的一阶逻辑有：

$$\{ T \subseteq S \rightarrow Th(T) \subseteq Th(S) \}$$

$$\text{假设 } T \vdash \text{fly}(\text{tweety}) \quad (2.15)$$

$$\text{并且 } S = T \cup \{ \neg \text{fly}(\text{tweety}) \} \quad (2.16)$$

现在，因为 $\text{fly}(\text{tweety})$ 是属于 T ，则由(2.15)和(2.16)有

$$S \vdash \text{fly}(\text{tweety}) \quad (2.17)$$

$$\text{根据 } S \text{ 的定义（见（2.16）），有 } S \vdash \neg \text{fly}(\text{tweety}) \quad (2.18)$$

表达式(2.17)和(2.18)显然是矛盾的，因此 $Th(T) \subseteq Th(S)$ 不满足。这说明不能用 \vdash 代替 \vdash 。

如何对 \vdash 进行解释呢？假设将含有模态词 \Diamond 的一阶谓词演算系统记为 FC ，将允许使用 \Diamond 的一阶公式的全体集合记为 L_{FC} ，则对任何公式集合 $\Gamma \subseteq L_{FC}$ ， $Th(\Gamma)$ 的意义为：

$$Th(\Gamma) = \{ A \mid \Gamma \vdash_{FC} A \}$$

为了更清楚地定义 $Th(\Gamma)$ ，首先定义非单调算子 NM_{Γ} 。它可基于 Γ 和相容性操作符 \Diamond 定义如下：

对任意的公式集合 $S \subseteq L_{FC}$ ，

$$NM_{\Gamma}(S) = Th(\Gamma \cup ASM_{\Gamma}(S))$$

其中 $ASM_{\Gamma}(S)$ 称为 S 的假设集：

$$ASM_{\Gamma}(S) = \{ \Diamond Q \mid Q \in L_{FC} \wedge \neg Q \notin S \}$$

那么可以定义 $Th(\Gamma)$ 为：

$$Th(\Gamma) = \cap (\{ L_{FC} \} \cup \{ S \mid NM_{\Gamma}(S) = S \})$$

从上式可以看出 $Th(\Gamma)$ 可以说是 NM_{Γ} 算子的所有固定点的交集，而当 NM_{Γ} 无固定点时， $Th(\Gamma) = \cap (\{ L_{FC} \}) = \{ L_{FC} \}$ ，即约定 $Th(\Gamma)$ 为全体 FC 公式的集合。

那么 \vdash 可以理解为：如果 $P \in Th(\Gamma)$ ，那么称 P 可由 Γ 非单调地推出，并记为 $\Gamma \vdash P$ 。

注意算子 NM_{Γ} 有固定点时， $\Gamma \vdash P$ 是指 P 在算子 NM_{Γ} 的每一个固定点中，而在缺省理论中， P 在 Δ 中“可证”，是指 P 属于 Δ 的某一个扩张，即在算子的某一个固定点中。

例 2.10 假设为 Γ 公理理论，包括

$$\Diamond P \rightarrow \neg Q \text{ 以及 } \Diamond Q \rightarrow \neg P$$

形式地表示为： $\Gamma = FC \cup \{ \Diamond P \rightarrow \neg Q, \Diamond Q \rightarrow \neg P \}$

则该系统有两个固定点 $(P, \neg Q)$ 、 $(\neg P, Q)$ ，即一个固定点含有 $\neg Q$ ，不含有 $\neg P$ 。另一个固定点含有 $\neg P$ ，不含有 $\neg Q$ 。

而对于 $\Gamma = FC \cup \{ \Diamond P \rightarrow \neg P \}$ ，该系统没有固定点。

设 $NM_{\Gamma}(S) = S'$ ，若 $\neg P \notin S$ ，那么 $\Diamond P \in ASM_{\Gamma}(S)$ ，从而 $\neg P \in S'$ ；反之，若 $\neg P \in S$ ，则 $\Diamond P \notin ASM_{\Gamma}(S)$ ，故 $\neg P \notin S'$ 。这就是说 S 不可能等于 S' ， NM_{Γ} 无固定点。

上述现象可以解释如下：

$$\{ \Diamond P \rightarrow \neg Q, \Diamond Q \rightarrow \neg P \} \vdash (\neg P \vee \neg Q)$$

$$\{ \Diamond P \rightarrow \neg P \} \vdash \text{矛盾}$$

McDermott 和 Doyle 指出采用 NML 进行推理会有两个问题：

- (1) 无法从 $\Diamond(A \wedge B)$ 推出 $\Diamond A$
- (2) $\{\Diamond P \rightarrow Q, \neg Q\}$ 非单调推出的结论是什么？

为了克服上述问题，像相容性操作符 \Diamond 一样，McDermott 和 Doyle 借用了其他模态逻辑的操作符，称为必然性操作符，记为 \Box ，它们之间关系如下：

$$\Box P \equiv \neg \Diamond \neg P$$

$$\text{或者：} \quad \Diamond P \equiv \neg \Box \neg P$$

第一个定义是说：P 是必然的，可以用另外的方式表示为 P 的否定是不相容的。第二个定义是说：P 是相容的，可以表示为 P 的否定不是必然的。

2.8 自认知逻辑

2.8.1 Moore 系统 \mathcal{L}_B

自认知逻辑(autoepistemic logic)是 Moore 首先引进的[Moore 1985]，它的目的是要刻画智能主体对于自己的知识和信念进行推理的规律。从纯粹逻辑的角度看，自认知逻辑是一种具有一个模态算子 **B** 的模态逻辑，其中 **B** 被解释为“相信”或者“知道”。如果把一个主体的信念表示成一组逻辑公式，那么自认知逻辑的一个基本任务就是要刻画这组公式应该满足什么样的条件。直觉上，主体应该相信从它目前的信念出发利用逻辑规则能推导出来的那些事实，而且如果主体相信某个事实或者不相信某个事实，那么该主体应该相信它自己相信了某个事实或者相信它自己不相信某个事实。

Moore 考虑自认知理论 T 对于一组初始前提 A 是可靠的，当且仅当 T 中的每一个自认知解释器是一个 T 中的自认知模型，其中全部 A 的公式为真。一个理想的理性主体的信念必须满足下列条件：

- (1) 设 $P_1, \dots, P_n \in T$, and $P_1, \dots, P_n \vdash Q$, 则 $Q \in T$ 。
- (2) 设 $P \in T$, 则 $BP \in T$ 。
- (3) 设 $P \notin T$, 则 $BP \in T$ 。

在这种情况下，主体不能再得到更进一步的结论，因此，Moore 称上述理论为稳定自认知理论。当然，下列条件也成立：

- (4) 如果 $BP \in T$, 则 $P \in T$ 。
- (5) 如果 $BP \in T$, 则 $P \notin T$ 。

自认知逻辑的基本符号由可数多个命题符号，逻辑联结词 \neg , \wedge 以及模态词 **B** 构成。我们把这样一个逻辑记作为 \mathcal{L}_B 。

2.8.2 \mathcal{L}_B 逻辑

在 Moore 系统 \mathcal{L}_B 的基础上，Levesque 引入另一个模态词 **0**，希望表达“关于某个方面，某个事实是主体所知道的仅有的（所有的）情况”这样一类陈述，形成 \mathcal{L}_B 逻辑[Levesque

1990]。在 \mathcal{L}_B 和 $0\mathcal{L}$ 中，公式及其他的语法概念与一般的模态逻辑一样的方式定义。如果在公式 ψ 中不出现模态词 **B** 以及 **0** 我们就把这个公式称为一个客观的公式，否则就称为主观的公式。在 $0\mathcal{L}$ 中，如果公式 ψ 中不出现模态词 **0** 我们就把该公式称为基本公式。如果我们把每个形如 $B\psi$ 的公式看成一个命题符号，那么根据经典命题演算的有关结论，我们可以把每个自认知逻辑的公式化成（析取以及合取）标准型。如下就是所谓的 Moore 析取标准型定理，其合取标准型定理可类似地给出。

定理 2.6 (Moore 析取标准型定理) 对任意公式 $\psi \in \mathcal{L}_B$ ， ψ 等价于一个具有以下形状 $\psi_1 \vee \psi_2 \vee \cdots \vee \psi_k$ 的公式，这里每个 ψ_i ($1 \leq i \leq k$) 具有形状

$$B\varphi_{i,1} \wedge \cdots \wedge B\varphi_{i,m_i} \wedge \neg B\varphi_{i,1} \wedge \cdots \wedge \neg B\varphi_{i,n_i} \wedge \varphi_{ii}$$

ψ_{ii} 是客观公式。

下面，我们在 $0\mathcal{L}$ 中定义模型的概念以及模型和公式之间的可满足关系。不失一般性，我们假定在我们的可数多个命题符号集（记作 L ）上存在一个序关系。记 2^L 为所有的从 L 到 $\{0, 1\}$ 所有函数，即所有赋值的集合。在 $0\mathcal{L}$ 中，一个模型就是一个对偶 W, w ，这里 W 是 2^L 的一个子集，而 w 是 2^L 中的一个元素。

定义 2.20 在 \mathcal{L}_B 中，令 W, w 是一个模型，而 ψ 是一个公式。我们定义满足关系 $W, w \models \psi$ 如下：

- (1) 如果 ψ 是一个命题符号 p ，则 $W, w \models p$ 当且仅当 $w(p) = 1$ ；
- (2) $W, w \models \neg \psi$ 当且仅当 $W, w \not\models \psi$ ；
- (3) $W, w \models (\psi \wedge \varphi)$ 当且仅当 $W, w \models \psi$ 而且 $W, w \models \varphi$ ；
- (4) $W, w \models B\psi$ 当且仅当对所有的 $w' \in W$ ， $W, w' \models \psi$ 。

定义 2.21 令 W, w 是一个模型， φ 是 $0\mathcal{L}$ 中的一个公式， $W, w \models 0\varphi$ 当且仅当 $W, w \models B\varphi$ ，而且对每个 w' ，如果 $W, w' \models \varphi$ 则 $w' \in W$ 。

模态词 **0** 是通过对模态词 **B** 的定义作必要的修改而得到的。它们间的关系可从如下两式中看到

$$W, w \models B\psi \text{ 当且仅当对于每一个 } w', w' \in W \Rightarrow W, w' \models \psi;$$

$$W, w \models 0\psi \text{ 当且仅当对于每一个 } w', w' \in W \Leftrightarrow W, w' \models \psi.$$

引进了模态词 **0** 的好处在于，它和稳定扩张之间有着密切的联系。在一定程度上，我们可以通过它来刻画稳定扩张。以下的结论表明了这个事实。

定理 2.7 (稳定扩张) 对于每个基本公式 ψ 以及每个极大赋值集 W ， $W \models 0\psi$ 当且仅当

$\{\psi: \psi \text{ 是基本公式而且 } W \vdash B\psi\}$ 是 ψ 的一个稳定扩张。

推论 2.1 公式 ψ 的稳定扩张的数目与所有的满足 0ψ 的极大赋值集的数目相等。

2.8.3 标准型定理

标准型定理对于探索稳定集和稳定扩张的结构是十分重要的。如上文所述, Moore 利用经典命题逻辑的方法得到了两个自认知逻辑的标准型定理。我们是直接从语义的角度出发的, 而且可以看到我们得到的标准型是相当简单的。

定义 2.22 对于每个基本公式 ψ , 我们递归地定义它的秩 $\text{rank}(\psi)$ 如下:

- (1) 如果 ψ 是一个客观公式, 则 $\text{rank}(\psi) = 0$;
- (2) 如果 $\psi = \psi_1 \wedge \psi_2$, 则 $\text{rank}(\psi) = \text{Max}(\text{rank}(\psi_1), \text{rank}(\psi_2))$;
- (3) 如果 $\psi = \neg \varphi$, 则 $\text{rank}(\psi) = \text{rank}(\varphi)$;
- (4) 如果 $\psi = B\varphi$, 则 $\text{rank}(\psi) = \text{rank}(\varphi) + 1$ 。

引理 2.1 以下陈述成立:

- (1) $\vdash B(B(\psi)) \leftrightarrow B(\psi)$;
- (2) $\vdash B(\neg B(\psi)) \leftrightarrow \neg B(\psi)$;
- (3) $\vdash B(B(\psi) \wedge \varphi) \leftrightarrow B(\psi) \wedge B(\varphi)$;
- (4) $\vdash B(\neg B(\psi) \wedge \varphi) \leftrightarrow \neg B(\psi) \wedge B(\varphi)$;
- (5) $\vdash B(B(\psi) \vee \varphi) \leftrightarrow B(\psi) \vee B(\varphi)$;
- (6) $\vdash B(\neg B(\psi) \vee \varphi) \leftrightarrow \neg B(\psi) \vee B(\varphi)$ 。

引理 2.2 $\vdash B(B(\psi_1) \vee \dots \vee B(\psi_s) \vee \neg B(\varphi_1) \vee \dots \vee \neg B(\varphi_t) \vee \varphi) \leftrightarrow$
 $(B(\psi_1) \vee \dots \vee B(\psi_s) \vee \neg B(\varphi_1) \vee \dots \vee \neg B(\varphi_t) \vee B(\varphi))$ 。

定理 2.3 (合取标准型定理) 对于每个 $\psi \in L_B$, ψ (语义) 等价于形状如 $\psi_1 \wedge \psi_2 \wedge \dots \wedge \psi_k$ 的一个公式, 其中 ψ_i ($1 \leq i \leq k$) 具有形状 $B\varphi_{i,1} \vee \dots \vee B\varphi_{i,m_i} \vee \neg B\varphi_{i,1} \vee \dots \vee \neg B\varphi_{i,n_i} \vee \psi_{ii}$, 而 $\varphi_{i,j}$, $\varphi_{i,n}$ ($1 \leq i \leq k$, $1 \leq j \leq m_i$, $1 \leq n \leq n_i$, ψ_{ii} 是客观公式)。

证明 对公式 ψ 的秩进行归纳。如果 $\text{rank}(\psi) = 1$, 我们要证明的结论就是 Moore 的合取标准型定理。以下假设我们的结论在公式的秩不超过 $N-1$ 时都成立而且 $\text{rank}(\psi) = N$ 。根据 Moore 的析取标准型定理我们有

$$\psi = \psi^1 \vee \psi^2 \vee \dots \vee \psi^k$$

这里每个 ψ^i ($1 \leq i \leq k$) 具有形状

$$B\varphi_{i,1}^i \wedge \dots \wedge B\varphi_{i,m_i}^i \wedge \neg B\varphi_{i,1}^i \wedge \dots \wedge \neg B\varphi_{i,n_i}^i \wedge \psi_{ii}^i$$

根据归纳假设, $\text{rank}(\varphi_{i,j}^i) \leq N-1$, $\text{rank}(\varphi_{i,t}^i) \leq N-1$, $\text{rank}(\psi_{ii}^i) = 0$ 。

于是 $\varphi_{i,j}^i$, $\varphi_{i,t}^i$ 等价于一个秩至多为 1 的公式。不失一般性, 令 $\varphi_{i,j}^i$ 具有如下的形状

$$x^1 \wedge \dots \wedge x^d$$

这里每个 x_h ($1 \leq h \leq d$) 具有形状

$$Bx_{h,1} \vee \dots \vee Bx_{h,u_h} \vee \neg Bx'_{h,1} \vee \dots \vee \neg Bx'_{h,v_h} \vee x_{hh}$$

[$x_{h,j}$, $x'_{h,n}$ ($1 \leq h \leq d$, $1 \leq j \leq u_h$, $1 \leq n \leq v_h$), x_{hh} 是客观公式]。

于是根据语义定义, $B\varphi_{i,j}^i$ 等价于

$$B(x_1) \wedge \cdots \wedge B(x_d) \quad (2.17)$$

而且根据引理 2.2 每个 $B(x_h)$ 都等价于

$$Bx_{h,1} \vee \cdots \vee Bx_{h,u_h} \vee \neg Bx'_{h,1} \vee \cdots \vee \neg Bx'_{h,v_h} \vee Bx_{hh} \quad (2.18)$$

$[x_{h,j}, x'_{h,n} (1 \leq h \leq d, 1 \leq j \leq u_h, 1 \leq n \leq v_h), x_{hh}, \text{是客观公式}]$ 。

现在用式(2.18)来替代每个 Bx_h , 而用式(2.17)来替代每个 $\varphi_{i,j}^i$ 等等。于是我们就得到一个与 ψ_L -等价而且秩至多为 1 的公式。

利用经典命题逻辑一样的技巧把所得到的公式化成析取标准型, 最后我们就得到了我们希望得到的公式。

利用对偶性, 我们得到如下的推论。

推论 2.2 (析取标准型定理) 对于每个 $\psi \in L_B$, ψ_L -等价于一个具有以下形状的公式:

$\psi_1 \vee \psi_2 \vee \cdots \vee \psi_k$, 这里每个 $\psi_i (1 \leq i \leq k)$ 具有形状 $B\varphi_{i,1} \wedge \cdots \wedge B\varphi_{i,m_i} \wedge \neg B\varphi_{i,1} \wedge \cdots \wedge \neg B\varphi_{i,n_i} \wedge \psi_{ii}$, 而 $\varphi_{i,j}, \varphi_{m,n} (1 \leq i, m \leq k, 1 \leq j \leq m_i, 1 \leq n \leq n_i, \psi_{ii}$ 是客观公式。

2.8.4 \Diamond -记号以及稳定扩张的一种判定过程

这里, 我们首先给出 \Diamond -记号的定义。因为 2^L 是一个布尔代数而我们的模型是 2^L 中的子集以及元素构成的对偶, 因此如果把我们的逻辑的语义按照集合论的格式进行定义, 我们将得到一个关于自认知逻辑直观简洁的理解。同时我们也将得到一个关于稳定扩张的代数解释。基于以上的新解释我们构造我们的关于稳定扩张的判定过程。

定义 2.23 (\Diamond -记号) 对于每个基本公式 ψ , 我们定义关于 ψ 的 \Diamond -记号 \Diamond_ψ 如下:

- (1) 如果 ψ 是一个命题符号 p , 则 $\Diamond_p = \{w: w \in 2^L \text{ 而且 } w(p) = 1\}$;
- (2) 如果 $\psi = \neg\varphi$, 则 $\Diamond_{\neg\varphi} = \sim\Diamond_\varphi = 2^L - \Diamond_\varphi$, 这里, \sim 是集合论中的求补运算符号;
- (3) 如果 $\psi = \psi_1 \wedge \psi_2$, 则 $\Diamond_{\psi_1 \wedge \psi_2} = \Diamond_{\psi_1} \cap \Diamond_{\psi_2}$, 这里, \cap 是集合论中的求交集运算符号;
- (4) 如果 $\psi = B\varphi$, 则 $\Diamond_{B\varphi} = \Diamond_\varphi$ 。

引理 2.3 令 ψ, φ 是客观公式, 则

- (1) $\vdash \psi \rightarrow \varphi$ 当且仅当 $\Diamond_\psi \subseteq \Diamond_\varphi$;
- (2) $\{\psi, \varphi\}$ 可满足当且仅当 $\Diamond_\psi \cap \Diamond_\varphi \neq \emptyset$ 。

现在我们用集合论的格式来定义自认知逻辑的语义。根据我们的标准型定理, 只需要对其中的几种情形加以考虑。

定理 2.8 令 ψ, φ 是客观公式而 W, w 是一个模型, 则

- (1) $W, w \models \psi$ 当且仅当 $w \in \Diamond_\psi$;
- (2) $W, w \models \psi \wedge \varphi$ 当且仅当 $W, w \models \psi$ 而且 $W, w \models \varphi$;

(3) $W, w \vdash \neg B \psi$ 当且仅当 $W \notin \Diamond \psi$;

(4) $W, w \vdash B \psi$ 当且仅当 $W \subseteq \Diamond \psi$ 。

下面我们先来定义一种 **0-性质**，然后来看看在我们的意义之下算子 **0** 的含义。

定义 2.24 令 ψ 是一个基本公式并且具有定理 2.5 中所给出的标准型定理的形状。令 $J \subseteq \{1, \dots, k\}$ 。如果以下条件成立我们就说 J 具有 **0-性质**：

- (1) 对于每个 $r \in J$, $\bigcup_{j \in J} \Diamond \psi_{ji} \vdash B\varphi_{r,1} \wedge \dots \wedge B\varphi_{r,m_r} \wedge \neg B\varphi_{r,1} \wedge \dots \wedge \neg B\varphi_{r,m_r}$;
- (2) 对于每个 $t \notin J$, $\bigcup_{j \in J} \Diamond \psi_{jj} \vdash B\varphi_{t,1} \wedge \dots \wedge B\varphi_{t,m_t} \wedge \neg B\varphi_{t,1} \wedge \dots \wedge \neg B\varphi_{t,m_t}$;

我们可以从两方面来判定一个集公式是否具有 **0-性质**。第一种方法属于集合论的范畴，而第二种则属于逻辑语义的范畴。它们事实上是等价的。

引理 2.4 (集合论方法) J 具有 **0-性质** 当且仅当以下条件成立：对于每个 $r \in J$,

$1 \leq p_1 \leq m_r, 1 \leq p_2 \leq n_r, \Diamond_J \subseteq \Diamond \varphi_{r,p_1}$ 而且 $\Diamond_J \not\subseteq \Diamond \varphi_{r,p_2}$ 成立；而且对于每个 $t \notin J$, 存在 $1 \leq q_1$

$\leq m_t$ 或者 $1 \leq q_2 \leq n_t$ 使得 $\Diamond_J \not\subseteq \Diamond \varphi_{t,q_1}$ 或者 $\Diamond_J \subseteq \Diamond \varphi_{t,q_2}$, 这里 \Diamond_J 是 $\bigcup_{j \in J} \Diamond \psi_{jj}$ 的缩写。

引理 2.5 (语义方法) J 具有 **0-性质** 当且仅当以下的公式集是可满足的

$\{\psi^J \rightarrow \varphi_{r,p_1} : r \in J, 1 \leq p_1 \leq m_r\} \cup \{\neg \varphi_{r,p_2} : r \in J, 1 \leq p_2 \leq n_r\} \cup \{\psi^J\} \cup \{\bigvee_{t \notin J, 1 \leq q_1 \leq m_t} \{\psi^J \wedge \neg \varphi_{t,q_1}\} \vee \bigvee_{t \notin J, 1 \leq q_2 \leq n_t} \{\psi^J \rightarrow \varphi_{t,q_2}\}\}$,

这里 ψ^J 是 $\bigvee_{j \in J} \psi_{jj}$ 的缩写。根据标准型定理以及引理 2.4 和 2.5 以下定理显然成立。

定理 2.9 对于任何的基本公式 ψ , 在定义 2.13 的假设之下, 存在一个判定过程判定集合 $\{1, \dots, k\}$ 是否具有 **0-性质**。

为了判定 J 是否具有 **0-性质**, 我们只需要判定引理 2.5 是否是可满足的。因为任何一集(有限)的客观公式都可以作为引理 2.5 中的第二三个分支的候选对象, 因此根据 SAT 问题是 NP-完全问题, 判定 $\{1, \dots, k\}$ 的一个子集是否具有 **0-性质** 的问题是 NP-完全问题。

定理 2.10 在定义 2.24 的假设之下, 令 W, w 是一个模型。则 $W, w \vdash 0 \psi$ 当且仅当存在 $\{1, \dots, k\}$ 的具有 **0-性质** 的子集 J 使得 $W = \bigcup_{j \in J} \Diamond \psi_{jj} = \Diamond \bigcup_{j \in J} \psi_{jj}$ 。

证明 回顾定义, 我们有 $W \vdash 0 \psi$ 当且仅当

- (1) $W, w \vdash B \psi$;
- (2) 对于每个 w , 若 $W, w \vdash \psi$ 则 $w \in W$ 。

首先我们假设 $W, w \vdash 0 \psi$ 。令 $J = \{j : W, w \vdash B\varphi_{j,1} \wedge \dots \wedge B\varphi_{j,m_j} \wedge \neg B\varphi_{j,1} \wedge \dots \wedge \neg B\varphi_{j,m_j}\}$ 。于是根据(1)我们有 $W \subseteq \bigcup_{j \in J} \Diamond \psi_{jj}$; 而且根据(2)我们有 $\bigcup_{j \in J} \Diamond \psi_{jj} \subseteq W$ 。所以 $W = \bigcup_{j \in J} \Diamond \psi_{jj}$ 。根据 **0-性质** 我们容易验证另一个方向也是正确的。

推论 2.3 令 ψ 是基本公式, 则在定义 2.13 的假设之下, ψ 所具有的稳定扩张的个数与 $\{1, \dots, k\}$ 的具有 **0-性质** 的子集的个数相等。

推论 2.4 令 ψ 是基本公式, 则在定义 2.13 的有关假设之下, ψ 存在唯一的稳定扩张当且仅当存在唯一的 $J \subseteq \{1, \dots, k\}$ 使得 J 具有 **0-性质**。

推论 2.3 和推论 2.4 在具体地判定公式集是否具有稳定扩张方面十分的简洁直观。以下

我们看几个例子。

例 2.11 令 ψ 为 Bp 。于是 ψ 等价于 $Bp \wedge \neg B(r \wedge \neg r) \wedge (q \vee \neg q)$ 。此时唯一可能的极大集是 $\Diamond_{(q \vee \neg q)}$ ，即 2^L 。因此 ψ 没有稳定扩张。

例 2.12 令 ψ 为 p 。这时 p 等价于 $B(q \vee \neg q) \wedge \neg B(r \wedge \neg r) \wedge p$ ，因为 $\Diamond_p \subseteq \Diamond_{(q \vee \neg q)} = 2^L$ 而且

$\Diamond_p \not\subseteq \Diamond_{(r \wedge \neg r)} = \emptyset$ ，所以 p 具有唯一的稳定扩张。

例 2.13 令 ψ 为 $(\neg Bp \rightarrow q) \wedge (\neg Bq \rightarrow p)$ 。这时，我们的公式等价于 $(Bp \wedge Bq) \vee (Bp \wedge p) \vee (Bq \wedge q) \vee (p \wedge q)$ ，因此可能的具有 0-性质的极大集有： \Diamond_p ， \Diamond_q ， $\Diamond_{p \wedge q}$ 以及 $\Diamond_{p \vee q}$ 。容易验证只有 \Diamond_p 和 \Diamond_q 具有 0-性质。因此给定的公式集有两个稳定扩张。

由以上的一系列结果，我们有以下的关于稳定扩张的判定过程：

输入： 一个基本公式 ψ 。

初始状态： $N=0$ 。

第一步： 把 ψ 化成秩至多为 1 的析取标准型；

令 k 是 ψ 的上述析取标准型中析取分支的个数；

$2^k = 2^k$ 。

[这里 2^k 表示由 $\{1, \dots, k\}$ 的所有子集合构成的集合。]

第二步： 循环。

如果 $2^k \neq \emptyset$ ，任取 $J \in 2^k$ ；

如果 J 具有 0-性质， $N=N+1$ ， $2^k = 2^k - J$ ，

重复第二步；

如果 J 没有 0-性质， $2^k = 2^k - J$

重复第二步；

输出： N (ψ 的稳定扩张的个数)。

2.9 真值维护系统

真值维护系统 TMS(truth maintenance system)是一种与默认推理既有联系，又有区别的一种推理技术，正如一阶谓词和产生式系统既有联系又有区别一样。真值维护系统是大型推理系统的一个子系统，实现对知识库中信念(belief)的维护[Doyle 1979]。它的基本问题是：① 必须在不完全的、有限的信息基础上做出假设的决策，使该假设成为知识库的信念；② 当这些决策的结论被以后的事实证明为错误时，如何对其信念进行修正。

真值维护系统有两种基本的数据结构：结点表示信念和理由(justification)表示信念的原因。真值维护系统的理由与默认理论的理由含义不同。前者不仅包括已知的知识也包括假设的知识。TMS 的基本操作是：新结点的形成——将信念赋与该结点；一个结点的新理由的

加入——表示把某个信念与该结点联接起来。新的理由的加入给我们提供这样一种可能：对每个信念结点，把一个或多个理由作为它的充分支持。当该结点为不可信时(可能由于新知识的加入引起)真值维护系统就可据此进行信念修正。具体实现则是：寻找这样的结点，它的理由充足的支持依赖于已改变的信念，并修正这些结点的信念。另外由于要求每个结点都必须有理由充分的支持与之相联，这就保证了下述过程的实现——相关性回溯，通过跟踪矛盾信念的理由充足的支持，去掉引起该矛盾的假设结点之一以消除矛盾，并同时生成一个记录用来避免以后出现类似的矛盾。

真值维护系统的实现主要包括两个过程，一个是默认假设的形成，另一个是相关性回溯过程，它们均依赖于信念的表示方法。

1. 信念知识表示

一个结点可能有若干个理由，每个理由表示该结点中信念的一个原因。一个结点是可信的，如果它的理由至少有一个是有效的。所谓有效是指它可从现行知识库(包括假设的信念集)中推出。

在真值维护系统中，每一个命题或规则均称为结点，它分为两类：

IN-结点	相信为真
OUT-结点	不相信为真，或无理由相信为真， 或当前没有任何有效的理由。

这样，任何命题 p 就有四种知识状态：表示 p 为 IN-结点或 OUT-结点，及表示 p 为非 IN-结点和 OUT-结点。

每个结点附有理由表，表中每一项表示具体结点的有效性。在真值维护系统中有两类不同的理由表，一个称为支持表 SL(support list)，另一个称为条件证明 CP(conditional proof)。前者是它所在结点的信念之原因，即该信念的存在依赖于该表中的理由；而后者则是出现矛盾的原因，即一个矛盾结点的存在是该表中的理由所致。

支持表 SL 的形式：

$$(\{SL\}(\langle IN\text{-结点表}\rangle)(\langle OUT\text{-结点表}\rangle)) \quad (2.19)$$

这里 IN-结点表中的 IN-结点表示知识库中的已知知识，而 OUT-结点表中的 OUT-结点则表示这些结点的否定，不在知识库中，为默认知识。显然，如果 OUT-结点表为空，则该系统蜕化为单调推理。如果支持表 SL 中的 IN-结点表中每个结点当前都为 IN-结点，且在 OUT-结点表中每个结点当前都为 OUT-结点，那么支持表 SL 理由是有效的。

支持表 SL 最通用。例如：

- (1) 现在是夏天 (SL () ())
- (2) 天气很潮湿 (SL (1) ())

结点(1)的 SL 表中的 IN-结点表和 OUT-结点表为空，表明它不依赖于任何别的结点中当前的信念或默认信念。这类结点称为前提。而结点(2)的 SL 表中 IN-结点表含结点(1)，这说明导致结点(2)可信任结论的推理链依赖于当前在结点(1)的信念。由此可见，真值维护系统

的推理与谓词逻辑系统相类似，不同的是真值维护系统可以撤消前提，并可以对知识库作适当修改。

如果支持表 SL 中的 OUT-结点表不为空，如：

- (1) 现在是夏天 (SL () ())
- (2) 天气很潮湿 (SL (1) (3))
- (3) 天气很干燥

若结点(1)是 IN，结点(3)是 OUT，结点(2)才为 IN。这个证实表明：如果现在是夏天，又没有天气很干燥的证据，那么天气很潮湿。如果将来某一时刻出现了天气很干燥的证据，即为结点(3)提供了一个证据，则结点(2)就变为 OUT，因为它不再有一个有效的证实。像结点(2)这样的结点称为假设，它与非空的 OUT 结点表的 SL 证实有关。OUT 结点(3)是结点(2)之证实的一部分。但如果结点(3)不存在，就不能这样表示了。

在真值维护系统中，它仅利用证实来维持一个相容的信念数据库，真值维护系统本身并不产生证实。上面的证实必须由使用真值维护系统的问题求解程序提供。

条件证明 CP 的形式为：

$$(CP \langle \text{结论} \rangle \langle \text{IN-假设} \rangle \langle \text{OUT-假设} \rangle) \quad (2.20)$$

如果结论结点为 IN-结点，以及下列条件成立：

- (1) IN-假设中的每个结点都是 IN-结点；
- (2) OUT-假设中的每个结点都是 OUT-结点。

那么条件证明 CP 是有效的。一般说来，OUT-假设总是空集。真值维护系统要求假设集划分成两个不相交的子集，分别为不导致矛盾的假设和导致矛盾的假设。

条件证明 CP 的证实表示有前提的论点。一般只要在 IN-假设中的结点为 IN，OUT-假设中的结点为 OUT，则结论结点为 IN。于是，条件证明的证实有效。处理 CP 比 SL 更难。事实上，真值维护系统把它们转换为 SL 证实来处理。

2. 默认假设

令 $\{F_1, \dots, F_n\}$ 表示所有可能的候选的默认假设结点集，G 表示选择默认假设的原因的结点，即由 G 引起在 $\{F_1, \dots, F_n\}$ 中进行缺省选择。这样我们给结点 Node(F_i) 以如下理由：

$$(SL(G)(F_1, \dots, F_{i-1}, F_{i+1}, \dots, F_n)) \quad (2.21)$$

而选取 F_i 为默认假设。如果不存在任何其它关于如何进行选择的信息，则可认为除 F_i 之外其它任何候选都不是可信的。这样 F_i 为 IN，其它 $F_j (j \neq i)$ 均为 OUT。但如果接收到一个有效的理由支持某个其它候选 F_j ，则 F_j 就为 IN，而导致 F_i 的假设失败而变为 OUT。如果从假设 F_i 导出矛盾，则相关性回溯就根据对其它为 OUT 的候选的依赖性识别出 F_i 为一缺言假设，从而使 F_i 变为 OUT，而选取另外一个候选 F_j 。这样 F_j 的理由为：

$$(SL \langle \text{Variousthings} \rangle \langle \text{reminder} \rangle) \quad (2.22)$$

其中 reminder 表示除 F_i 和 F_j 之外的 F_k 集 ($1 \leq k \neq i, j \leq n$)。

如果在进行默认假设选择时，候选集不完整，而那些当前尚未出现的候选肯定会在以后

被发现，则上述方法就显得无能为力。对此，可采用下面的方法，它能逐渐扩大候选集。

保留上述的表示方法，令 F_i 为表示 F_i 的否定的新结点。如果在当前的知识库中不能证明 F_i ，则认为 F_i 是可信的，并建立适当的理由使得如果 F_j 不同于 F_i ，则 F_j 蕴含 F_i ，即 F_j 是 F_i 的理由。这样就必保证 F_i 的理由为：

$$(SL (G) (F_i)) \quad (2.23)$$

而 F_i 的一组理由则为：

$$(SL (F_j)) (j \neq i) \quad (2.24)$$

这样，如果没有充分理由必须使用其它候选，则就假设 F_i 为一信念。但若从这种默认假设中导出矛盾，则说明 F_j 的假设是错误的，从而证实 F_i ，使 F_i 结点为 IN-结点。 F_j 结点变为 OUT-结点，这时就必须使用相关性回溯来检查导致矛盾的原因，重新构造新的默认假设。

3. 相关回溯

TMS 的相关回溯是在知识库中出现不一致性时，寻找并删除已做的一个不正确的默认假设，恢复一致性。它包括下列三个步骤：

(1) 从产生的矛盾结点开始，回溯跟踪该矛盾结点的理由充足的支持以寻找矛盾的假设集，并从中去掉至少一个假设信念以消除矛盾。这个过程首先收集导致矛盾的“极大”假设集。所谓“极大”假设集是指理由充足的支持关系在结点上产生一个自然偏序。如果结点 N 出现在结点 M 的理由充足的支持中，那么结点 N 比结点 M “小”。而“极大”假设集则是在这种偏序关系上为极大结点的结点集。

(2) 构造一个结点记录矛盾产生的原因。这一步根据第一步收集到的“极大”假设集分析导致矛盾的原因。设 $S = \{A_1, \dots, A_n\}$ 表示不一致假设集，则回溯过程生成一个称为 nogood 的结点，它表示为：

$$A_1 \wedge \dots \wedge A_n \rightarrow \text{false}$$

$$\text{或等价地 } \neg (A_1 \wedge \dots \wedge A_n) \quad (1)$$

$$\text{其理由为：} (CP \langle \text{矛盾结点} \rangle S \langle \rangle) \quad (2)$$

这样即使在去掉了某个假设从而消除了矛盾之后，这个假设集的不一致性仍被记录了下来。

(3) 从 S 中选取假设 A_i (即不合理假设)，并证实列在其理由充足的支持条件中的一个 OUT-结点。令 NG 为 nogood 结点， s 同前， D_1, \dots, D_k 是出现在支持 A_i 的信念的理由条件中，则使用下面的理由条件

$$(SL (NG A_1 \dots A_{i-1} A_{i+1} \dots A_n) (D_2 \dots D_k)) \quad (3)$$

表证实 D_1 ，从而使 A_i 的理由条件失败。只要 nogood 和其它假设是可信的，而其它不合理假设的否定不是可信的，那么理由条件(3)是有效的。如果 A_i 选择有误，则在以后就会产生另一个包含 D_1 的矛盾。如果在加入了(3)之后现行矛盾仍然没有消除，则重新回溯直到发现矛盾消除为止。

这里，以找一个时间安排一次会议为例，说明相关回溯的工作过程。首先假设日期为

星期三:

(1) 会议日期为星期三 (SL () (2))

(2) 会议日期不应是星期三

目前没有相信“会议日期不应是星期三”的证实,所以结点(1)是 IN,以表示会议日期为星期三这一假设。根据与会者的时间,安排会议系统通过推理得到会议必须在 2:00 举行的结论,这是根据若干结点得到的。这样,真值维护系统具有如下知识库:

(1) 会议日期为星期三 (SL () (2))

(2) 会议日期不应是星期三

(3) 会议时间为 14:00 (SL (32, 40, 61) ())

接着,安排会议的程序要找一间会议室,结果发现星期三下午两点钟无空会议室,于是产生下一个结点通知 TMS:

(4) 矛盾 (SL (1, 3) ())

这时,调用相关回溯,查看矛盾结点 SL 的证实表中的结点,比如说 $A_1 \cdots A_k$,然后向后跟踪,通过 A_i 的 SL 证实中的结点,比如说 $B_1 \cdots B_s$,再回到 B 的 SL 证实中的结点,寻找假设。试图找到这样一个假设集,只要删除该集中的某个假设,则矛盾就可消除。在此例中,这个集只包含一个元素,即结点(1)。回溯机制通过产生一个不相容结点来记录它。

(5) 不相容 (CP 4 (1, 3) ())

现在真值维护系统选择不相容假设中的一个,即结点(1),使它的 OUT 表中的一个结点变为 IN 来使结点(1)变为 OUT。于是有:

(2) 会议日期不应是星期三 (SL (5) ())

结点(2)与结点(5)为 IN,就引起结点(1)为 OUT,因为结点(1)的证实依赖于结点(2)是 OUT。结点(4)现在也变成 OUT。这样一来,矛盾就消除了,可选择一个新的日期。由于矛盾中不包含时间,所以会议时间仍为 2:00 不变。

Doyle 在他的文章里探讨了这种非单调推理的形式化和理论问题。de Kleer 分析了这种系统的不足之处,并设计了基于假设的真值维护系统称为 ATMS (assumption-based TMS) [de Kleer 1986]。该系统允许各种互相对立的假设和信念同时存在,克服了真值维护系统的一些重要缺点。

ATMS 由两部分组成,一个问题求解器,一个 TMS。前者包含了领域的所有知识和推理过程,每个推理结果都传送给 TMS。TMS 的工作则是在目前给定的理由条件下判断哪些知识是可信的,哪些是不可信的。问题求解器向 ATMS 每次提供一个理由条件和假设,ATMS 构造一个直接的数据结构进行快速的相容性检查。TMS 在每一时刻只能处理一组假设,而 ATMS 考虑假设集合。ATMS 论证规则的形式为:

$$A_1, a_2, \cdots, a_n \Rightarrow D$$

表示:若 A_1, a_2, \cdots, a_n 皆成立,则 D 亦成立。其中, a_1, a_2, \cdots, a_n 称为前提,它们构成的集合称为论据, D 称为结论。前提和结论都是结点,但都不能是恒假结点,不许带“非”符号。当前

提为空时规则成为：

$$\Rightarrow D$$

表示 D 不依赖于任何前提而成立，即为恒真结点。

限于篇幅，这里略去 ATMS 的详细讨论，读者可参阅有关文献。

2.10 情景演算的逻辑基础

在计算机科学的许多领域，“动作”（或称为行为、操作等等）是一个基本的概念。例如在数据库理论中，对数据记录的“删除”、“插入”以及“修改”等等就是人们对数据库经常采取的一些操作（动作）。这些操作对数据库来说无疑是最基本的而且也是最重要的。再如在分布式人工智能中，具有多个自主的主体的系统中，主体之间是通过它们的各式行为（动作）来相互沟通和相互影响的；在多主体系统中，一个重要的研究方向就是研究主体的知识和信念。在这方面的研究中“行为”的概念是最基本的，它对于研究主体知识和信念的获得以及修改等问题都是很关键的。与“动作”的概念相伴随的是所谓“进化”的概念。对于诸如数据库中对数据的某种操作以及多主体系统中某个主体的一个行为这样的动作，对于数据库和主体的信念都产生一定的效用，其结果就是数据库的更新和主体知识和信念的修改，也就是所谓数据库的进化和主体知识和信念的进化。

深入研究涉及动作和进化等概念的问题，对于计算机科学具有普遍的重要意义。情景演算是分析处理和研究涉及动作和进化问题的最常用的形式工具。在一般的利用情景演算分析处理涉及动作和进化概念的文献中，大量的分析处理是建立在对情景演算的直观理解的意义上的。Reiter 和林方真以数据库理论为背景，建立了一个多类逻辑（我们称之为 **LR**），并且把情景演算的概念和方法在这种特殊的多类一阶逻辑的框架之内进行描述，以便为有关的研究提供一个坚实的系统的理论基础[Lin 1994]。把情景演算集成在一个多类逻辑框架里，这一做法的核心是：为了刻画一个动作，只需要描述动作发生的条件和动作发生以后对其环境所产生的效果这两件事。为此，在逻辑框架 **LR** 中引入了“动作”，“状态”和“一般对象”这三种个体类型，然后通过一系列的逻辑句子来表述这三种对象的最一般关系以及动作发生的前提和后果。每个这样的句子集被称为一个基本的动作理论。从纯粹逻辑学的观点看，所谓的“动作的基本理论”就是在特定的多类逻辑中的普通逻辑学意义下的一个理论。

我们在对 **LR** 进行分析的基础上建立了一个多类逻辑，作为描述动作的框架[田启家 1997]。为了使动作的概念得到恰当的描述，我们不把动作当作为一种个体类型对待，而是把它作为一类函数加以处理。这样做既符合对于动作的直观理解同时又使得它在逻辑框架里具有清晰的语义。这里着重分析了所谓的极小动作理论，给出了其模型的直观表示，定义和分析了进化的概念，并结合数理逻辑的有关方法得到了一些关于极小动作理论的进化的可定义性方面的结论。

研究情景演算的逻辑基础，不仅仅是计算机科学的需要，对它的研究对于逻辑学的也具

有重要的价值。由于一般多类逻辑的类型之间缺乏明确的直观意义，一般地，逻辑学家们认为，可以通过引入若干一元谓词的方法把多类逻辑的研究归结为经典一阶逻辑的研究。

2.10.1 刻划情景演算的多类逻辑

LR 被定义成一种多类逻辑，在其形式语言 \mathcal{L} 中引入了三种关于个体的类型，即状态类型 s 、对象类型 o 和动作类型 a ，一个类型为 s 的常量符号 S_0 (表示起始状态)，一个类型为 $\langle a, s; s \rangle$ 的二元函数符号 do (描述一个动作的发生使得状态从一个变成另外一个)，一个类型为 $\langle a, s \rangle$ 的二元关系符号 $Poss$ (表示一个动作在一个状态之下是可能发生的)，和一个类型为 $\langle s, s \rangle$ 的二元关系符号 $<$ (表示状态之间的先后关系)。

\mathcal{L} 中的关系符号如果其参数的类型均为 o ，则称该关系符号是独立于状态的关系符号； \mathcal{L} 中的函数符号如果其自变量和因变量的类型均为 o ，则称该函数符号是独立于状态的函数符号； \mathcal{L} 中的关系符号如果其参数的类型有且仅有一个是 s ，而其余的类型均为 o ，则称该关系符号是一个动态关系。LR 假定 \mathcal{L} 由有限多个常量符号、状态独立的关系符号和函数符号、有限多个动态关系符号以及上段特别列出来的形式符号构成。

LR 中的语法概念如项、原子公式及公式等模仿一般多类逻辑的类似定义。依照习惯， \mathcal{L} 也表示此时所有一阶多类逻辑公式的集合。

给定一个状态类型的项 st ，定义 \mathcal{L}_{st} 和 \mathcal{L}_{st}^2 如下：

\mathcal{L}_{st} 中的公式是 \mathcal{L} 中公式的一个子集，其每个成员中除 st 外不包含其它类型为 s 的项，不对状态变元使用量词，不包含关系符号 $Poss$ 和 $<$ 。形式地它是满足以下条件的最小的公式集：

(1) 如果 $\psi \in \mathcal{L}$ 不含有任何状态类型的项，那么 $\psi \in \mathcal{L}_{st}$ ；

(2) 对于每个有 n 个操作对象的动态关系 F 以及类型 o 的项 x_1, \dots, x_n ， $F(x_1, \dots, x_n, st) \in \mathcal{L}_{st}$ ；

(3) 如果 $\psi, \varphi \in \mathcal{L}_{st}$ ，那么就有 $\neg\psi, \psi \wedge \varphi, \psi \vee \varphi, \varphi \rightarrow \psi, \psi \leftrightarrow \varphi, \forall x \psi, \exists x \psi, \forall a \psi$

以及 $\exists a \psi$ 都属于 \mathcal{L}_{st} ，这里 x, a 分别具有类型 o 和 a 。

\mathcal{L}_{st}^2 定义为满足以下条件的最小公式集：

(1) $\mathcal{L}_{st} \subseteq \mathcal{L}_{st}^2$ ；

(2) 对于每个类型 \mathbf{o} 的论域上的 n 元的谓词变元 p , 以及类型为 \mathbf{o} 的项 \vec{x} , $p(\vec{x}) \in \mathcal{L}_{st}^2$;

(3) 如果 $\psi, \varphi \in \mathcal{L}_{st}^2$, 那么就有 $\neg\psi, \psi \wedge \varphi, \psi \vee \varphi, \varphi \rightarrow \psi, \psi \leftrightarrow \varphi, \forall p \psi, \exists p \psi, \forall x \psi,$

$\exists x \psi, \forall a \psi$ 以及 $\exists a \psi$ 都属于 \mathcal{L}_{st} , 这里 x, a 分别具有类型 \mathbf{o} 和 \mathbf{a} , p 是类型 \mathbf{o} 的论域上的的谓词变元。

2.10.2 LR 中的基本动作理论

在 LR 中, 一个基本的动作理论 \mathbf{D} 就是一个如下的一组句子:

$$\mathbf{D} = \Sigma \cup \mathbf{D}_{ss} \cup \mathbf{D}_{ap} \cup \mathbf{D}_{una} \cup \mathbf{D}_{s0} \quad (2.25)$$

其中

(1) Σ 是一组逻辑公式。它的直观含义是说所有类型为 \mathbf{s} 的个体在二元关系 $<$ 之下构成一个分叉的时序结构, 在其中 S_0 是起点, 而 do 是后继函数。而且所谓的树型归纳法也包含在其中。具体地它包含如下的公理:

$$\begin{aligned} S_0 &\neq do(a, s); \\ do(a_1, s_1) &= do(a_2, s_2) \rightarrow (a_1 = a_2 \wedge s_1 = s_2); \\ \forall P [(P(S_0 \wedge \forall a, s (P(s) \rightarrow P(do(a, s)))) \rightarrow \forall s P(s)); \\ \neg (s < S_0); \\ s < do(a, s') &\leftrightarrow (Poss(a, s) \wedge s \leq s'). \end{aligned}$$

(2) \mathbf{D}_{ss} 是一组逻辑公式。它们描述了一个类型为 \mathbf{a} 的个体按一定的规则被执行以后所产生的效果。一般地, 其中的公式具有以下的形状:

$$Poss(a, s) \rightarrow (F(\vec{x}, do(a, s)) \rightarrow \psi_F(\vec{x}, a, s)) \quad (2.26)$$

这里 F 是一个动态关系符号而 ψ_F 是 \mathcal{L}_s 中的一个逻辑公式。

(3) \mathbf{D}_{ap} 是一组逻辑公式。它们描述了一个类型为 \mathbf{a} 的个体能够按一定的规则被执行的前提条件。它们一般具有以下的形状:

$$Poss(A(\vec{x}, s) \rightarrow \psi_A(\vec{x}, s)) \quad (2.27)$$

这里 A 是的一个动作常量, 而 ψ_A 是 \mathcal{L}_s 中的一个逻辑公式。

(4) \mathbf{D}_{una} 是表达以下意义的逻辑公式: 两个动作常量在两组类型为 \mathbf{o} 的个体上执行的结果是不同的, 除非它们分别是同一个动作以及同一个个体对象的序列。它们具有如下的形状:

$$\begin{aligned} A(\vec{x}) &\neq A'(\vec{y}); \\ A(\vec{x}) = A(\vec{y}) &\rightarrow (\vec{x} = \vec{y}). \end{aligned}$$

(5) \mathbf{D}_{s0} 是一组有限的 \mathcal{L}_{s0} 中的逻辑公式。称为基本动作理论的初始条件。

2.11 框架问题

当我们试图用形式逻辑的方法表示动作或事件的影响的时候，困难之一就是框架问题。如前面所述，框架问题就是如何表示和推演出那些在动作执行后不会发生改变的性质或事实。该问题最早由 McCarthy 和 Hayes 提出。当某个动作执行或者某个事件发生时，如果我们使用古典逻辑来描述什么发生了变化，我们也必须描述什么没有发生变化。否则，我们使用这些描述得不到任何有用的结论。

当使用一阶逻辑来描述动作的影响时，则没有发生变化情况的描述要远远大于发生了变化情况的描述。当我们描述动作的影响时，我们应该能够关注于什么发生了变化，也能够想到什么没有发生变化。因此所谓框架问题就是建立一个形式系统来处理上述情况的问题。

2.11.1 框架公理

虽然从 $\Delta \wedge \Sigma$ 中可以得到很多的结论，但是仍然有一些我们希望的结果无法得到。例如，在 S_0 中，B 在 C 的上面，移动 A 到 D 并没有改变这个事实，但是我们并没有：

$$\Delta \wedge \Sigma \models \text{Holds}(\text{On}(\text{B}, \text{C}), \text{Result}(\text{Move}(\text{A}, \text{D}), \text{S}_0))$$

也就是说，我们可以得到一个动作作用后，什么发生了改变，但是并不能得到什么没有发生改变。一般来说，动作只是具有“局部的”影响，很多的流并没有发生改变。为了对那些没有受到动作的影响，持续不变的流进行推理，需要为每个流和动作增加一些框架公理。

例如，对流 On，有框架公理：

$$\text{Holds}(\text{On}(v, w), \text{Result}(\text{Move}(x, y), s)) \leftarrow \text{Holds}(\text{On}(v, w), s) \wedge x \neq v$$

该框架公理是说，在移动 x 到 y 上之前， v 是在 w 上，则移动以后 v 仍然在 w 之上（只要 v 不是被移动的积木）。这样，加上 Δ 就可以得到 $\text{Holds}(\text{On}(\text{B}, \text{C}), \text{Result}(\text{Move}(\text{A}, \text{D}), \text{S}_0))$ 。对于流 Clear，具有类似的框架公理：

$$\text{Holds}(\text{Clear}(x), \text{Result}(\text{Move}(y, z), s)) \leftarrow \text{Holds}(\text{Clear}(x), s) \wedge x \neq z$$

下面向积木世界中增加一个新的动作和新的流。

- $\text{Color}(x, c)$ ：表示积木 x 具有颜色 c 这一事实。
- $\text{Paint}(x, c)$ ：表示在积木 x 上涂上颜色 c 这一动作。因为涂色总能够成功，因此没有前提条件。

这样在 Δ 中又会增加下面的公式：

$$\text{Holds}(\text{Color}(x, c), \text{Result}(\text{Paint}(x, c), s))$$

假设情景 S_0 的描述和 Σ 中是一样的，只是所有的积木都是红色的：

$$\text{Holds}(\text{Color}(x, \text{Red}), \text{S}_0)$$

令 Δ' 和 Σ' 分别表示新的域描述和新的 S_0 描述。移动一个积木并不会改变该积木的颜色，并且如果没有其他的公理就无法得到下面的结论：

$$\Delta' \wedge \Sigma' \models \text{Holds}(\text{Color}(\text{A}, \text{Red}), \text{Result}(\text{Move}(\text{A}, \text{D}), \text{S}_0))$$

因此需要增加下面两个公理：积木的颜色在移动动作下保持不变；积木的颜色不受其他积木是否涂色的影响。

$$\text{Holds}(\text{Color}(x, c), \text{Result}(\text{Move}(y, z), s)) \leftarrow \text{Holds}(\text{Color}(x, c), s)$$

$$\text{Holds}(\text{Color}(x, c1), \text{Result}(\text{Paint}(y, c2), s)) \leftarrow \text{Holds}(\text{Color}(x, c1), s) \wedge x \neq y$$

除了上面的公理外，动作 Paint 也不会对 On 和 Clear 流产生影响：

$$\text{Holds}(\text{On}(x, y), \text{Result}(\text{Paint}(z, c), s)) \leftarrow \text{Holds}(\text{On}(x, y), s)$$

$$\text{Holds}(\text{Clear}(x), \text{Result}(\text{Paint}(y, c), s)) \leftarrow \text{Holds}(\text{Clear}(x), s)$$

至此，我们得到了一个由九个公式和六个框架公理组成的域描述。框架公理被用来证明如果状态由一个不影响属性的动作改变，那么这个属性保持不变。一般来说，大多数动作不会对大多数流产生影响，每次增加一个新的流都需要增加和域中动作数大致一样多的新的框架公理，并且每次增加一个新的动作，都需要增加和域中流的数量大致一样多的框架公理。或者说，在一个域中，如果有 n 个流和 m 个动作，则总的框架公理数为： $n \times m$ [Shanahan 1997]，这样用情景演算的方式来表达动作如何改变世界就会变得很难管理。

有人就如何减少大量的框架公理数进行了研究。问题是即使能够减少大量的框架公理，用它们对关于在几个动作序列上什么流不会改变的推理的计算仍然是笨重的。

在古典、一阶逻辑中，在不对情景演算做大的改变的情况下，也可以找到一种简洁的方法来表示框架公理中的信息。例如，前面所给出的框架公理都有如下的形式：

$$\text{Holds}(f, \text{Result}(a, s)) \leftarrow \text{Holds}(f, s) \wedge \Pi$$

其中 f 是一个流， a 是一个动作， Π 是一个合取式。这样，就可以采用所谓的通用框架公理表示上述的框架公理形式：

$$\text{Holds}(f, \text{Result}(a, s)) \leftarrow \text{Holds}(f, s) \wedge \neg \text{Affects}(a, f, s)$$

这样，对每个原有的框架公理，需要增加一个负原子 Affects 。例如对于积木世界，有：

$$\begin{aligned} \neg \text{Affects}(\text{Move}(x, y), \text{On}(v, w), s) &\leftarrow x \neq v \\ \neg \text{Affects}(\text{Paint}(z, c), \text{On}(x, y), s) & \\ \neg \text{Affects}(\text{Move}(y, z), \text{Clear}(x), s) &\leftarrow x \neq z \\ \neg \text{Affects}(\text{Paint}(y, c), \text{Clear}(x), s) & \\ \neg \text{Affects}(\text{Move}(y, z), \text{Color}(x, c), s) & \\ \neg \text{Affects}(\text{Paint}(y, c2), \text{Color}(x, c1), s) &\leftarrow x \neq v \end{aligned}$$

根据这些公理，可以推出和前面一样的结论。虽然每个公理比前面给出的公理形式上简洁了一些，但是现在的主要问题是公理的数量问题。如果再对上述公理进行整理，则还可以减少公理的数量。

$$\begin{aligned} \text{Affects}(a, \text{On}(x, z), s) &\rightarrow a = \text{Move}(x, y) \\ \text{Affects}(a, \text{Clear}(x), s) &\rightarrow a = \text{Move}(x, y) \\ \text{Affects}(a, \text{Color}(x, c2), s) &\rightarrow a = \text{Paint}(x, c1) \end{aligned}$$

上述这些公理就是所谓的解释闭包公理（explanation closure axioms）。解释闭包公理是框架公理的代换，但是更加地简洁。它们构成了框架问题的单调解决方案的基础。

2.11.2 框架问题解决方案的准则

怎样以一种形式化的、逻辑的方法表示动作的影响，而不用写出所有的框架公理？这就是框架问题所要研究的问题。这里我们需要考虑：对框架问题的一个解决方案，什么是该方案可接受的准则？在 [Shanahan 1997] 中，作者给出了三个准则：

- 表示的简洁性（representational parsimony）
- 表达的灵活性（expressive flexibility）
- 精细的容错性（elaboration tolerance）

对于表示的简洁性，这是框架问题的本质，它是说对动作影响的表示应该是简洁的。由于无法对该简洁性给出准确的量化，一个可行的方案是表示的大小大致是和论域复杂度成正比，论域复杂度一个比较好的指标是动作的总数量加上流的总数量。

一个有效的框架问题解决方法需要能够应用于广泛的涉及表示的问题，对于一个复杂的

领域，简洁的表示方法还有很多问题需要解决，因此框架解决方案还需要满足第二个准则：表达的灵活性。这里复杂的领域并不仅仅是说有更多数量的动作和流，更主要的是指这些领域有其特殊的性质，在表示它们之前可能需要进行深入地思考和研究。例如，它们可能具有下面的特性：

- 分枝 (ramification)
- 并发动作
- 非确定动作
- 连续动作

如果我们考虑领域的限制的话，一个动作除了它的直接影响外，还有一些分枝。领域的限制（有时称为状态限制）是指在同一情景下流的什么样的组合可以保持不变。例如，我们只有三个积木，一个放置在另一个上面。令流 $\text{Stack}(x, y, z)$ 表示积木 x 、 y 、 z 为一个堆。可以像对流 On 、 Clear 、 Color 一样给出流的效应公理，不过这里用下面的公式通过领域限制来表示堆的效应公理：

$$\text{Holds}(\text{Stack}(x, y, z), s) \leftarrow x \neq \text{Table} \wedge \text{Holds}(\text{On}(y, x), s) \wedge \text{Holds}(\text{On}(z, y), s)$$

在我们至今所考虑的积木世界的例子中，没有出现两个动作同时执行的情况。实际上并发的动作或事件每天都在发生。例如当有多个人来移动积木的时候，就会有同时发生的动作。使框架问题难于处理的另一个复杂领域特性是非确定性动作或动作的影响并不完全知道。复杂领域的再一个特性是连续动作的发生。显然，连续动作也是到处存在的，如行进中的汽车、填充容器等等。积木世界的情景演算表示目前为止都是离散的。这样表示并不是说所表示出的改变都是离散的，而是便于抽象。然而，用离散的方法进行抽象并不总是合适的，有的领域中有些量的连续变化是其很重要的特性，需要用连续的方法进行表示。连续变化在情景演算中非常难于表示，这是研究事件演算的动机之一。在事件演算中，无需太多的困难就可以表示连续的动作。

最后一个评价准则是精细的可接受程度。一个表示是精细可接受的，是指当向表示中增加新的信息的时候，所付出的努力是和信息的复杂程度成正比的。如果扩充一个情景演算理论，增加一个动作时会影响到 n 个流，可能会需要增加 n 个句子，但是这并不需要完全地重新构造原有的理论。受新的动作所影响的事实需要逐渐地加入到原有的理论中。

但是对于一个动作具有分枝等性质的领域来说，仅仅简单地扩充解释闭包公理是不够的，需要重新地构造，对原有系统做大的改进。在单调性机制下重新构造解释闭包公理以融合新的动作和流是很困难的，需要大量的计算。因此，如果我们在单调性领域中处理框架问题，看来是无法满足精细程度的要求，因此可以说无法用单调的方法来处理框架的问题。其根本原因就是单调性机制下必须以框架公理或解释闭包公理的形式对动作执行后没有发生变化的性质给出“显式”的描述。为解决这一问题，人工智能研究者提出了非单调推理机制，其中最常用的推理机制包括 McCarthy 的限制理论和 Reiter 缺省逻辑。可以说，框架问题的研究是促使进行非单调推理研究的动力之一。用非单调机制解决框架问题的核心思想在于形式化地刻画如下一条常识性原则：惯性原则 (law of inertia)，即在典型的情况下，一个性质在动作执行后不发生变化的。

在大多数情况下，人们一般只是写出效应公理，而对于其他情况会缺省地认为“没有发生改变”，然后借助某些非单调机制来推出结论。从二十世纪八十年代早期以来，已经有了一些非单调机制用来处理这种问题，最常用的有限制理论和缺省逻辑。另外，如果语义定义的合适的话，逻辑程序设计的否定看作失败也可以用来处理这种问题。

2.11.3 框架问题的非单调解决方案

最先用非单调机制解决框架问题的是 McCarthy，他提出了著名的限制理论，并尝试用限制理论来解决框架问题。限制理论能够说明一些谓词的扩展是最小的。例如从公式集 Γ 可以推出 $P(A)$ ，但是除了 A 外，对其他所有的 x ，我们并不能证明 $P(x)$ 或者 $\neg P(x)$ 。这是由于 Γ 最小化 P 的限制，记作 $CIRC[\Gamma; P]$ ，即对所有的 x ，除非 Γ 说明 $P(x)$ 为真，否则都为假。例如，从 $CIRC[\Gamma; P]$ 可以得到 $\neg P(B)$ 。

显然使用限制理论解决框架问题的方法是最小化谓词 $Affects$ 。这就是 McCarthy 所建议的方法，不过他定义的是谓词 Ab 而不是 $Affects$ 。但是最小化 $Affects$ 会产生违背直觉的结果。最著名的例子就是所谓的耶鲁射击（Yale Shooting）问题。

根据 McCarthy 和 Hayes 以及 Sandewall 的观点：解决框架问题的关键是对惯性原则的形式化，惯性原则是指：

惯性是正常的，变化是异常的

对惯性原则的形式化包含对下面缺省规则的形式化。

正常情况下，对于任何动作（或事件）和任何流，动作并不会对该流有所影响。

考虑前面提到的通用框架公理，由于有了负信息，因此框架公理需要考虑到负和正的信息的不变性。即通用框架公理应该表示一个流应该具有和动作执行前一样的值（正或负），除非动作影响到了流，即该公理是说如果一个动作没有影响到流，则在该动作执行前后，该流不变。

$$F1: [Holds(f, Result(a, s)) \leftrightarrow Holds(f, s)] \leftarrow \neg Affects(a, f, s) \quad (2.28)$$

这样就留给了我们复杂的工作：准确地说明哪个流没有受到哪个动作的影响。这也是框架问题的本质。

用 McCarthy 建议的 Ab 谓词代替 $Affects$ 得到：

$$F2: [Holds(f, Result(a, s)) \leftrightarrow Holds(f, s)] \leftarrow \neg Ab(a, f, s) \quad (2.29)$$

假设，用 Σ 表示效应公理、领域限制和观察到的句子的总和。扩充 Σ 以包含惯性原理的简单方式是结合 (F2) 然后限制它，极小化 Ab 并允许 $Holds$ 变化。换句话说，研究 $CIRC[\Sigma \wedge (F2); Ab; Holds]$ 。

极小化 Ab 并允许 $Holds$ 变化的限制策略看来似乎能够解决框架问题。然而，McDermott 和 Hanks 在 1968 年的工作表明，极小化 Ab 并允许 $Holds$ 变化这种方法即使对于非常简单的例子可能也得不到我们所需要的结论。他们把这种困难用一个简单的例子来说明，这就是著名的耶鲁射击问题。

在耶鲁射击问题中，某人被一枪打死了。McDermott 和 Hanks 的形式化包含

■ 三个动作：Load、Wait 和 Shoot。

■ 两个流：Alive、Loaded。

■ 两个效应公理：Load 动作往枪里填装了子弹；只要枪里有子弹，在 Shoot 动作后，受害人死了。

$$Y1: Holds(Loaded, Result(Load, s)) \quad (2.30)$$

$$Y2: \neg Holds(Alive, Result(Shoot, s)) \leftarrow Holds(Loaded, s) \quad (2.31)$$

■ 另外还有两个观察到的句子：在开始受害人是活着的；枪是没有子弹的。

$$Y3: Hold(Alive, S0) \quad (2.32)$$

$$Y4: \neg Holds(Loaded, S0) \quad (2.33)$$

下面首先定义一个谓词：UNA，表示名字的唯一性（uniqueness-of-name）。

UNA($f1, f2, \dots, fk$)表示：

对所有的 $i < j < k$: $f_i(x_1, x_2, \dots, x_m) \neq f_j(y_1, y_2, \dots, y_n)$, 并且

对所有的 $i < k$: $f_i(x_1, x_2, \dots, x_n) = f_i(y_1, y_2, \dots, y_n) \rightarrow [x_1=y_1 \wedge x_2=y_2 \wedge \dots \wedge x_n=y_n]$

这样, 我们有:

$$Y5: \text{UNA}[\text{Load}, \text{Wait}, \text{Shoot}] \quad (2.34)$$

$$Y6: \text{UNA}[\text{Alive}, \text{Loaded}] \quad (2.35)$$

$$Y7: \text{UNA}[S0, \text{Result}] \quad (2.36)$$

现在考虑动作序列 Load、Wait、Shoot 之后的情景会是什么, 或者说情景 $\text{Result}(\text{Shoot}, \text{Result}(\text{Wait}, \text{Result}(\text{Load}, S0)))$ 会是什么。如果将限制策略应用于这些公式, 哪些流会成立呢? 直觉上看, 在 Load 动作之后, 枪里有了子弹, 在 Wait 动作之后, 枪里还是有子弹, 即 Loaded, 由于枪里有子弹, 因此在 Shoot 动作之后, 受害人就死了。因此我们希望由 (Y1) 到 (Y7) 的限制可以得到下面的结论:

$$\neg \text{Holds}(\text{Alive}, \text{Result}(\text{Shoot}, \text{Result}(\text{Wait}, \text{Result}(\text{Load}, S0)))) \quad [2.37]$$

但是[2.37]并不能从这些限制中推出。

命题 2.1 (Hanks-McDermott 问题) 如果 Σ 是由 (Y1) 到 (Y7) 组成的, 则有

$$\text{CIRC}[\Sigma \wedge (F2); \text{Ab}; \text{Holds}] \models$$

$$\neg \text{Holds}(\text{Alive}, \text{Result}(\text{Shoot}, \text{Result}(\text{Wait}, \text{Result}(\text{Load}, S0))))$$

本命题的证明思路如下:

考虑满足下面条件的 $\Sigma \wedge (F2)$ 的模型 M:

$$M \models \text{Holds}(\text{Loaded}, \text{Result}(\text{Load}, S0))$$

$$M \models \neg \text{Holds}(\text{Loaded}, \text{Result}(\text{Wait}, \text{Result}(\text{Load}, S0))))$$

$$M \models \text{Holds}(\text{Alive}, \text{Result}(\text{Shoot}, \text{Result}(\text{Wait}, \text{Result}(\text{Load}, S0))))$$

显然这种模型是存在的, 并具有下面的性质:

$$M \models \text{Ab}(\text{Load}, \text{Loaded}, S0)$$

$$M \models \text{Ab}(\text{Wait}, \text{Loaded}, \text{Result}(\text{Load}, S0))$$

$$M \models \neg \text{Ab}(\text{Shoot}, \text{Alive}, \text{Result}(\text{Wait}, \text{Result}(\text{Load}, S0))))$$

考虑谓词 Ab 和 Hold, 若删除上述其中任何一性质, 都无法得到仅仅满足上述条件和性质的模型, 在这些模型中, 存在仅满足上述条件和性质的模型是最小模型。可能会存在多个最小的模型。由于在这些模型中有:

$$M \models \text{Holds}(\text{Alive}, \text{Result}(\text{Shoot}, \text{Result}(\text{Wait}, \text{Result}(\text{Load}, S0))))$$

这样也会有

$$\text{CIRC}[\Sigma \wedge (F2); \text{Ab}; \text{Holds}] \models$$

$$\neg \text{Holds}(\text{Alive}, \text{Result}(\text{Shoot}, \text{Result}(\text{Wait}, \text{Result}(\text{Load}, S0))))$$

所以说 McCarthy 的方法推不出期望的结果。Hanks 与 McDermott 的工作指出一般非单调性推理机制直接用于处理框架问题也都不是合适的。其原因在于这些机制中没有表达时间的概念, 时间的引入可以构成一个显示的次序, 而类似耶鲁射击这类问题的时间推理本质上是依赖这种次序的。

Hanks 与 McDermott 采用次序极小化方法处理框架问题。该方法的基本思想是尽可能地推迟状态变化的发生, 越晚越好, 只有在非发生不可的情况下才发生变化。

Hanks 与 McDermott 问题是由于时间方向性而产生的。当使用缺省逻辑来处理框架问题时该问题就成为主要的问题。前面我们已经对默认推理进行了介绍, 这里为了便于后面的说明, 重新给出去一些定义。

定义 2.24 缺省理论就是 $\langle \Delta, \Sigma \rangle$ 对。其中 Δ 是缺省规则集合, Σ 是一阶谓词演算的子句。

Σ 包含了领域知识，从中可以进行有效的演绎推理。 Δ 表示缺省知识，从中可以得到缺省结论。这里所考虑的缺省理论 $\langle \Delta, \Sigma \rangle$ 为：

$$\Delta = \left\{ \frac{\neg Ab(a, f, s)}{\neg Ab(a, f, s)} \right\}$$

而 Σ 是由上一小节的(Y1)到(Y4)和(F2)组成。

定义 2.25 如果合式公式是缺省理论 $\langle \Delta, \Sigma \rangle$ 关于操作符 Γ 的固定点（定义如下），则该合式公式是缺省理论 $\langle \Delta, \Sigma \rangle$ 的扩张。如果 S 是没有自由变量合式公式集合，则 $\Gamma(S)$ 是最小的这样的集合：

- $\Sigma \subseteq \Gamma(S)$
- 如果 ϕ 是 $\Gamma(S)$ 的逻辑结论，则 $\phi \in \Gamma(S)$ ，并且
- 如果 Δ 包括缺省规则，

$$\frac{\phi_1(\bar{x}) : \phi_2(\bar{x})}{\phi_3(\bar{x})}$$

并且 $\phi_1(\tau_1 \dots \tau_n) \in \Gamma(S)$ ， $\neg \phi_2(\tau_1 \dots \tau_n) \notin \Gamma(S)$ ，则 $\phi_3(\tau_1 \dots \tau_n) \in \Gamma(S)$ ，其中每个 τ_i 都是没有变量的项。

每个缺省理论扩张该理论可接受的信念的集合。扩张的方法是很自然的：初始情况下，扩张集合仅仅包括 Σ 及其结论，然后重复选择可用的缺省规则，增加其结论，并形成相应的演绎闭包，直到再没有可以增加的结论。

详细地说，给定缺省理论 $\langle \Delta, \Sigma \rangle$ ，可以基于下面的算法来构造扩张。由于在这里我们只有规范缺省规则，所以可以具有下述的算法：

$S' = \{ \}$

$S = \Sigma \cup \Sigma$ 的逻辑结论

While $S \neq S'$

$S' := S$

选择任意的 ϕ_1 、 ϕ_2 、 ϕ_3 以及 τ_1, \dots, τ_n ，有

$$\frac{\phi_1(\bar{x}) : \phi_2(\bar{x})}{\phi_3(\bar{x})} \text{ 属于 } \Delta$$

并且 $\phi_1(\tau_1 \dots \tau_n) \in S$ ， $\neg \phi_2(\tau_1 \dots \tau_n) \notin S$

$S = S \cup \phi_3(\tau_1 \dots \tau_n)$

$S = S \cup S$ 的逻辑结论

End While

由于缺省规则可以应用无限次，因此对于很多情况，上述算法并不终止。但是，在这种情况下，每个中间 S 也是某种扩张的子集，所以仍然可以使用上述算法通过执行有限次的循环获得一些有用的信息。

下面我们把上述算法应用于缺省理论，以表示耶鲁射击问题。

开始时，集合 $S = \Sigma \cup \Sigma$ 的逻辑结论，选择下面的缺省规则：

$\neg Ab(\text{Wait}, \text{Loaded}, \text{Result}(\text{Load}, S0))$

由于它可 S 不矛盾，所以它是可行的。则由(F2)，增加：

$\text{Holds}(\text{Loaded}, \text{Result}(\text{Wait}, \text{Result}(\text{Load}, S0)))$

因此由(Y2):

$\neg \text{Holds}(\text{Alive}, \text{Result}(\text{Shoot}, \text{Result}(\text{Wait}, \text{Result}(\text{Load}, S_0))))).$

上述算法本可以继续下去，但是我们已经得到了扩张，该扩张包含了我们所需要的结论。上述扩张是一种有目的、计划好的扩张。下面讨论不规则的 (anomalous) 扩张，即当应用缺省规则的时候，和上面的选择不一样。开始时，集合 $S = \Sigma \cup \Sigma$ 的逻辑结论。但是现在应用缺省规则时，选择：

$\neg \text{Ab}(\text{Shoot}, \text{Alive}, \text{Result}(\text{Wait}, \text{Result}(\text{Load}, S_0)))$

该缺省规则和 S 是一致的，因此可以增加到 S 中。现在增加逻辑结论：

$\neg \text{Holds}(\text{Loaded}, \text{Result}(\text{Wait}, \text{Result}(\text{Load}, S_0)))$

我们已经知道：

$\text{Holds}(\text{Loaded}, \text{Result}(\text{Load}, S_0))$

因此有：

$\text{Ab}(\text{Wait}, \text{Loaded}, \text{Result}(\text{Load}, S_0))$

这样我们就得到了不希望的结果：

$\text{Holds}(\text{Alive}, \text{Result}(\text{Shoot}, \text{Result}(\text{Wait}, \text{Result}(\text{Load}, S_0))))).$

可见，缺省逻辑和限制理论一样对耶鲁射击问题都会产生不规则的扩展。也就是说 Hanks 与 McDermott 问题并不仅仅是限制理论的问题。另外，上面的扩展也说明为什么会出现这样的问题。当有次序地应用缺省规则的时候，就是有目的的扩展，也就是说最早的异常最先考虑。当缺省规则以相反的次序应用的时候，就是不规则的扩展，也就是后面的异常优先考虑。或者说，有目的的扩展是把变化尽可能推迟发生。这就是所谓的次序极小化方法。

次序极小化可以解决耶鲁射击问题的不规则扩展，并可以得到所需要的结论[朱朝晖 2001]。这是一个进步，但是耶鲁射击问题仅仅是一个例子，对于其他问题是否会有 Hanks 与 McDermott 问题或类似的问题出现呢？这是很重要的问题，确实有很多例子可以看到次序极小化也会得到和直觉相佐的结论。著名的例子就是所谓的丢车问题 (stolen car)。

耶鲁射击问题可以看作是一种时序投影问题 (projection problem)，或者是预测问题 (prediction problem)。在这类问题中，我们知道在初始状态下，什么流成立，并且我们想知道在一系列动作之后，什么流成立。投影问题是一种在时间上向前的推理，是从因到果的推理。

丢车问题一般被看作解释问题 (explanation problem)。在解释问题中，我们已知某些流在某些状态下成立，但是在初始状态并不成立，我们想知道是什么使得这些流成立，因此解释问题是在时间上向后的推理，是结果到可能的原因的推理。

丢车问题是指：一个人早上把车停在停车场并去工作，根据常识，根据惯性原则，中午时该车应该还在停车场，但是当晚上该人去停车场时，发现车不见了。现在需要找出车不见了的原因。唯一合理的解释就是该车在早上到晚上之间被偷走了。因此以前的缺省假设：中午时车还在停车场就值得怀疑了。

丢车问题可以表示成以下三个句子。这里采用两个连续的 Wait 动作，而不是显式地表示从早到晚这一时间区间。唯一的流是 Stolen，表示汽车不在停车场。

SC1: $\neg \text{Holds}(\text{Stolen}, S_0)$ (2.38)

SC2: $S_2 = \text{Result}(\text{Wait}, \text{Result}(\text{Wait}, S_0))$ (2.39)

SC3: $\text{Holds}(\text{Stolen}, S_2)$ (2.40)

这可能并不是表示丢车问题最好的方法，但是它可用来说明次序极小化的难点，它给出了一个很强的结论，即汽车是在第二个 Wait 期间丢失的。这里首先给出两个公理（考虑情景空间为树形结构）：

$\text{Arb1: Result}(a_1, s_1) = \text{Result}(a_2, s_2) \rightarrow a_1 = a_2 \wedge s_1 = s_2$ (2.41)

$$\text{Arb2: } S0 \neq \text{Result}(a, s) \quad (2.42)$$

那么由(SC1)到(SC3)和(Arb1)、(Arb2)、(F2)可以得到:

$$\text{Ab}(\text{Wait}, \text{Stolen}, \text{Result}(\text{Wait}, S0)) \wedge \neg \text{Ab}(\text{Wait}, \text{Stolen}, S0)。$$

实际上汽车也可以是在第一个 Wait 之后丢失的, 即我们希望有:

$$\text{Ab}(\text{Wait}, \text{Stolen}, \text{Result}(\text{Wait}, S0)) \vee \text{Ab}(\text{Wait}, \text{Stolen}, S0)。$$

成立, 并且存在模型 M1、M2 有

$$M1 \models \text{Ab}(\text{Wait}, \text{Stolen}, \text{Result}(\text{Wait}, S0))$$

$$M2 \models \text{Ab}(\text{Wait}, \text{Stolen}, S0)$$

解决 Hanks 和 McDermott 问题的另一种方法是因果极小化 (Causal minimization) 方法 [Shanahan 1997][朱朝晖 2001]。在因果极小化中, 所有的改变都是由动作引起的。这种方法用显式表达的理由刻画如下的思想: 任意流的值发生变化当且仅当有成功执行的动作引起它发生改变, 即任何改变都要有依据。

在该方法中, Lifschitz 在系统中引入了新的三元谓词 $\text{Cause}(a, f, v)$ 表示如行动 a 成功执行, 则流 f 取值为 v 。例如, 对于耶鲁射击问题, 有 $\text{Cause}(\text{Shoot}, \text{Alive}, \text{false})$, Shoot 的前提为 Loaded。Lifschitz 对 Cause 进行限定 (允许 Holds 变化), 由于 Cause 不含情景变元, 对其进行限定时不会引起 McCarthy 方法中在不同情景下的冲突。另外, 对 Cause 进行限定意味着模型中动作产生的效果越少模型越优先, 在耶鲁射击问题中, 对行动 Wait 没有显示地效果说明, 利用对 Cause 的限定就保证了 Wait 在优先模型中不会引起任何流的变化, 所以 Lifschitz 方法可以解决耶鲁射击问题。

2.12 动态描述逻辑 DDL

2.12.1 描述逻辑

描述逻辑是一种基于对象的知识表示的形式化, 也叫概念表示语言或术语逻辑。它是一阶逻辑的一个可判定的子集, 具有合适定义的语义, 并且具有很强的表达能力。一个描述逻辑系统包含四个基本组成部分: 表示概念和关系的构造集; TBox 包含断言; ABox 实例断言; TBox 和 ABox 上的推理机制。一个描述逻辑系统的表示能力和推理能力取决于对以上几个要素的选择以及不同的假设 [Baader 2003]。

描述逻辑中有两个基本元素, 即概念和关系。概念解释为一个领域的子集; 关系则表示在领域中个体之间所具有的相互关系, 是在领域集合上的一种二元关系。

在一定领域中, 一个知识库 $K = \langle T, A \rangle$ 由两个部分组成: TBox T 和 ABox A 。其中 TBox 是一个关于包含断言的有限集合, 也称为术语公理的集合。包含断言的一般形式为 $C \sqsubseteq D$, 其中 C 和 D 都是概念。ABox 是实例断言的有限集合, 形为 $C(a)$, 其中 C 是一个概念, a 是一个个体的名字; 或者形为 $P(a, b)$, 其中 P 为一个原始关系, a, b 为两个个体的名字。

一般地, TBox 是描述领域结构的公理的集合, 它具有两方面的作用, 一是用来引入概念的名称, 二是声明概念间的包含关系。引入概念名称的过程即可以表示为 $A \doteq C$ 或者 $A \sqsubseteq C$, 其中 A 即为引入的概念。概念间的包含关系的断言可以表示为 $C \sqsubseteq D$ 。对于概念定义和包含关系, 有: $C \doteq D \Leftrightarrow C \sqsubseteq D$ 且 $C \sqsupseteq D$ 。

ABox 是实例断言的集合, 用于指明个体的属性或者个体之间的关系。它有两种形式的断言, 一是指明个体与概念间的属于关系, 二是指明两个个体之间所具有的关系。在 ABox 中, 对于论域中任意个体对象 a 和概念 C , 关于对象 a 是否为概念 C 中的元素的断言称之为概念实例断言, 简称概念断言。若 $a \in C$, 则记为 $C(a)$; 若 $a \notin C$, 则记为 $\neg C(a)$ 。

对于两个对象 a, b 和关系 R , 如果 a 和 b 满足关系 R , 则称 $a R b$ 为关系实例断言, 表示为 $R(a, b)$ 。关系断言是用来指明两个对象之间所满足的基本关系或者对象的属性, 构成二元关系。

一般地, 描述逻辑依据提供的构造算子, 在简单的概念和关系上构造出复杂的概念和关系。通常描述逻辑至少包含以下算子: 交(\sqcap), 并(\sqcup), 非(\neg), 存在量词(\exists)和全称量词(\forall)。这种最基本的描述逻辑称之为 **ALC**。在 **ALC** 的基础上再添加不同的构造算子, 则构成不同表达能力的描述逻辑。例如若在 **ALC** 上添加数量约束算子 “ \leq ” 和 “ \geq ”, 则构成描述逻辑 **ALCN**, 这里不做详细介绍。**ALC** 的语法和语义如表 2.1 所示。

ALC 语义将概念解释为一定领域的子集, 关系是该领域上的二元关系。形式上, 一个解释 $I = (\Delta^I, \cdot^I)$ 由解释的领域 Δ^I 和解释函数 \cdot^I 所构成, 其中解释函数把每个原子概念 A 映射到 Δ^I 的子集, 而把每个原子关系 P 映射到 $\Delta^I \times \Delta^I$ 的子集。

- 一个解释 I 是包含断言 $C \sqsubseteq D$ 的模型, 当且仅当 $C^I \subseteq D^I$;
- 解释 I 是 $C(a)$ 的模型, 当且仅当 $a \in C^I$; I 是 $P(a, b)$ 的模型, 当且仅当 $(a, b) \in P^I$;
- 解释 I 是知识库 K 的模型, 当且仅当 I 是 K 中每个包含断言和实例断言的模型;
- 若 K 有模型, 则称 K 是可满足的;
- 若断言 δ 对于 K 的每个模型是满足的, 则称 K 逻辑蕴含 δ , 记为 $K \models \delta$ 。
- 对概念 C , 若 K 有一个模型 I 使得 $C^I \neq \emptyset$, 则称 C 是可满足的。知识库 K 中的概念 C

的可满足性可以逻辑表示为 $K \models C \sqsubseteq \perp$ 。

表 2.1 **ALC** 的语法和语义

构造算子	语法	语义	例子
原子概念	A	$A^I \subseteq \Delta^I$	Human
原子关系	P	$P^I \subseteq \Delta^I \times \Delta^I$	has-child

顶部	\top	Δ^I	True
底部	\perp	Φ	False
交	$C \sqcap D$	$C^I \cap D^I$	Human \sqcap Male
并	$C \sqcup D$	$C^I \cup D^I$	Doctor \sqcup Lawyer
非	$\neg C$	$\Delta^I - C^I$	\neg Male
存在量词	$\exists R. C$	$\{x \mid \exists y, (x, y) \in R^I \wedge y \in C^I\}$	\exists has-child. Male
全称量词	$\forall R. C$	$\{x \mid \forall y, (x, y) \in R^I \Rightarrow y \in C^I\}$	\forall has-child. Male

关于描述逻辑中的基本推理问题，主要包括概念的可满足性、概念的包含关系、实例检测、一致性检测等，其中概念的可满足性问题是最基本的问题，其它的推理基本上都可以转化为概念的可满足性问题。

在描述逻辑中，可以利用下述性质对推理问题进行约简，转化为概念的可满足性问题，进而将推理问题进行简化。对于概念 C, D ，有如下命题成立：

- (i) $C \sqsubseteq D \Leftrightarrow C \sqcap \neg D$ 是不可满足的；
- (ii) $C \doteq D$ 是等价的 $\Leftrightarrow (C \sqcap \neg D)$ 与 $(D \sqcap \neg C)$ 都是不可满足的；
- (iii) C 与 D 是不相交的 $\Leftrightarrow C \sqcap D$ 是不可满足的。

2.12.2 动态描述逻辑的语法

由于动态描述逻辑是在传统描述逻辑的基础上扩充得到的[Shi 2005]，而传统描述逻辑有很多种类，本节以最小的描述逻辑 ALC 为基础来研究动态描述逻辑 DDL。

定义 2.26 在 DDL 的语言中包括以下基本符号：

- ◆ 概念名： C_1, C_2, \dots ；
- ◆ 关系名： R_1, R_2, \dots ；
- ◆ 个体常元： a, b, c, \dots ；
- ◆ 个体变元： x, y, z, \dots ；
- ◆ 概念运算： \neg, \sqcap, \sqcup 以及量词： \exists, \forall ；
- ◆ 公式运算： $\neg, \wedge, \rightarrow$ 以及量词 \forall ；
- ◆ 动作名： A_1, A_2, \dots ；
- ◆ 动作构造：如 $;$ (合成)， \cup (交替)， $*$ (反复)， $?$ (测试)；
- ◆ 动作变元： α, β, \dots ；
- ◆ 公式变元： $\varphi, \psi, \pi, \dots$ ；
- ◆ 状态变元： u, v, w, \dots 。

定义 2.27 在 DDL 中，概念定义如下：

- (1) 原子概念 P 、全概念 \top 、和空概念 \perp 都是概念；
- (2) 如果 C 和 D 是概念，则 $\neg C$, $C \sqcap D$, $C \sqcup D$ 都是概念；
- (3) 如果 R 为关系， C 为概念，则 $\exists R. C$, $\forall R. C$ 都是概念；
- (4) 如果 C 是概念， α 是动作，则 $[\alpha]C$ 也是概念。

定义 2.28 DDL 的公式定义如下，其中 C 为任意概念， R 为关系， a, b 为个体常元， x, y 为个体变元， α 是动作：

- (1) 形如 $C(a)$, $R(a, b)$ 和 $[\alpha]C(a)$ 的表达式称为断言公式，它们是不带变元的；
- (2) 形如 $C(x)$, $R(x, y)$ 和 $[\alpha]C(x)$ 的表达式称为一般公式，它们是带变元的；
- (3) 断言公式和一般公式都是公式；
- (4) 如果 φ 和 ψ 是公式，则 $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \rightarrow \psi$, $\forall x\varphi$ 都是公式；
- (5) 如果 φ 是公式，则 $[\alpha]\varphi$ 也是公式。

定义 2.29 形如 $\{a_1/x_1, \dots, a_n/x_n\}$ 的有穷集合称为一个实例代换，其中 a_1, \dots, a_n 为个体常元，称为代换项， x_1, \dots, x_n 为个体变元，称为代换基，它们满足 $x_i \neq x_j$, $i, j \in \{1, \dots, n\}$, $i \neq j$ 。

定义 2.30 设 φ 为一公式， x_1, \dots, x_n 为出现在 φ 中的个体变元， a_1, \dots, a_n 为个体常元，令 φ' 为 φ 通过实例代换 $\{a_1/x_1, \dots, a_n/x_n\}$ 而得到的公式，则称 φ' 为公式 φ 的实例公式。

定义 2.31 DDL 中条件 (condition) 定义如下，其中 N_C 表示个体常元的集合， N_X 表示个体变元的集合， N_I 是 N_C 和 N_X 的并，即 $N_I = N_C \cup N_X$ ：

$\forall C, C(p), R(p, q), p=q, p \neq q$ 。其中 $p, q \in N_I$, C 是 DDL 的概念， R 是 DDL 的关系。

定义 2.32 一个动作描述是一个形如 $A(x_1, \dots, x_n) \equiv (P_A, E_A)$ 的表达形式，其中：

- (1) A 为动作名：指示动作表示符；
- (2) x_1, \dots, x_n 为个体变元，指定动作的操作对象，因此也称之为操作变元；
- (3) P_A 为前提公式集 (*pre-conditions*)，指定动作执行前必须满足的前提条件，即 $P_A = \{con \mid con \in condition\}$ ；
- (4) E_A 为结果公式集 (*post-conditions*)，指定动作执行后得到的结果集， E_A 是序对 *head* / *body* 的集合，其中 *head* = $\{con \mid con \in condition\}$ ，*body* 是一个条件。

说明：(1) 动作定义了状态间的转换关系，即一个动作 A 将一个状态 u 转换成状态 v ，如果在状态 u 下应用动作 A 则产生状态 v 。这种转换关系依赖于状态 u , v 是否分别满足动作 A 的前提公式集 (*pre-conditions*) 和结果公式集 (*post-conditions*)，记作 $u T_A v$ 。

(2) 因为动作 A 发生以前的状态也可以影响动作 A 的结果，因而前提公式与结果公式在描述上有些不同。对于结果公式 *head* / *body*，如果 *head* 中的每个条件在状态 u 中满足，则 *body* 中的每个条件在状态 v 中满足。

(3) 当动作 A 为空时，这意味着动作在任何状态都是可执行的，则 A 表示了静态的规则

$head / body$, 从此可以看出 DDL 中动作还可以用于描述领域的约束规则。或者说, 给定一条 $head \leftarrow body$ 形式的规则, 我们可以用一个动作来表示:

$$A(x_1, \dots, x_n) \equiv (\phi, \{head / body\}).$$

定义 2.33 设 $A(x_1, \dots, x_n) \equiv (P_A, E_A)$ 为一个动作描述, $A(a_1, \dots, a_n)$ 是在 $A(x_1, \dots, x_n)$ 上经过实例代换 $\{a_1/x_1, \dots, a_n/x_n\}$ 而得到的, 则称 $A(a_1, \dots, a_n)$ 为 $A(x_1, \dots, x_n)$ 的动作实例, 并称 $A(a_1, \dots, a_n)$ 为原子动作, $P_A(a_1, \dots, a_n)$ 称为动作 $A(a_1, \dots, a_n)$ 的前提集, $E_A(a_1, \dots, a_n)$ 称为动作 $A(a_1, \dots, a_n)$ 的结果集。

定义 2.34 DDL 的动作定义如下:

- (1) 原子动作 $A(a_1, \dots, a_n)$ 是动作;
- (2) 如果 α 和 β 为动作, 则 $\alpha; \beta$, $\alpha \cup \beta$, α^* 都是动作;
- (3) 如果 φ 为断言公式, 则 $\varphi?$ 也是动作。

2.12.3 动态描述逻辑的语义

动态描述逻辑 DDL 的语义是由以下各部分组成的一个结构:

1. 非空集合 Δ , 是 DDL 形式系统中所讨论的所有个体对象的集合, 称为论域;
2. 非空集合 \mathcal{W} , 是 DDL 形式系统中所有状态的集合, 称为状态集;
3. 一类被称之为解释的映射 I , 它对 DDL 中的个体常元、概念和关系加以解释:
 - (i) 个体常元是论域 Δ 中一个元素;
 - (ii) 概念是论域 Δ 的子集;
 - (iii) 关系是该论域上的二元关系;
4. 在状态集 \mathcal{W} 之上的二元关系 A 称之为动作, 它对状态集 \mathcal{W} 之上的转换关系进行解释。

下面分别进行语义解释。首先, 对于 DDL 中的一个状态 u , 该状态下的解释 $I(u) = (\Delta, \bullet^{I(u)})$ 由论域 Δ 和解释函数 $\bullet^{I(u)}$ 所构成, 其中解释函数把每个原子概念映射到 Δ 的子集, 把每个原子关系映射到 $\Delta \times \Delta$ 的子集。概念和关系的语义表示如下:

- 全概念 \top 的语义为论域 Δ , 即 $\top^{I(u)} = \Delta$;
- 空概念 \perp 的语义为空集 \emptyset , 即 $\perp^{I(u)} = \emptyset$;
- 若 C 为概念, 则 $C^{I(u)} \subseteq \Delta$;
- 若 R 为关系, 则 $R^{I(u)} \subseteq \Delta \times \Delta$;
- $(\neg C)^{I(u)} = \Delta - C^{I(u)}$;
- $(\neg R)^{I(u)} = \Delta \times \Delta - R^{I(u)}$;

- $(C \sqcap D)^{I(u)} = C^{I(u)} \cap D^{I(u)}$;
- $(C \sqcup D)^{I(u)} = C^{I(u)} \cup D^{I(u)}$;
- $(\exists R. C)^{I(u)} = \{x \mid \exists y, (x, y) \in R^{I(u)} \wedge y \in C^{I(u)}\}$;
- $(\forall R. C)^{I(u)} = \{x \mid \forall y, (x, y) \in R^{I(u)} \Rightarrow y \in C^{I(u)}\}$;
- $([\alpha]C)^{I(u)} = \{x \mid uT_{\alpha}v, x \in (C)^{I(v)}\}$ 。

由于个体常元并不依赖于一定的状态，因此本文采用个体常元的刚性原理，规定个体常元的命名都是唯一的，且不随状态的变化而变化。因此在下面的语义解释中，把个体常元在不同状态的解释 $a^{I(u)}$ 都简记为 a 。

下面对 DDL 中的条件进行语义解释。给定动作 $A(x_1, \dots, x_n) \equiv (P_A, E_A)$ ， N_X^A 是发生在 A 中所有变量的集合， $I = (\Delta^I, \cdot^I)$ 是一个解释，映射 $\gamma: N_X^A \rightarrow \Delta^I$ 是对动作 A 中变量的赋值。对于 DDL 的 ABox 中的个体常元 $a \in N_C$ ， \cdot^I 将 a 解释为 Δ^I 中的一个元素，即 $a^I \in \Delta^I$ 。为了简便，对于个体变量或个体常元 $p \in N_I$ ，定义它们的解释如下：

$$p^{I, \gamma} = \begin{cases} \gamma(p), & \text{如果 } p \in N_X \\ p^I, & \text{如果 } p \in N_C \end{cases}, \text{ 则 DDL 的条件的语义解释为:}$$

- 如果 $C^I = \Delta^I$ ，则 I 和 γ 满足条件 $\forall C$;
- 如果 $a^{I, \gamma} \in C^I$ ，则 I 和 γ 满足条件 $C(a)$;
- 如果 $a^{I, \gamma} = b^{I, \gamma}$ ，则 I 和 γ 满足条件 $a = b$;
- 如果 $a^{I, \gamma} \neq b^{I, \gamma}$ ，则 I 和 γ 满足条件 $a \neq b$;
- 如果 $\langle a^{I, \gamma}, b^{I, \gamma} \rangle \in R^I$ ，则 I 和 γ 满足条件 $R(a, b)$ 。

在一定状态 u 下，DDL 的断言公式将个体常元同概念和关系进行关联，它们可以分为两类，形如 $C(a)$ 的断言公式称为概念断言，形如 $R(a_1, a_2)$ 的断言公式称为关系断言。对于概念断言，它们用来说明一定状态下某个个体常元与某概念之间的关系，即元素与集合之间的关系，其语义解释为：

- $u \models C(a) \text{ iff } a \in C^{I(u)}$;
- $u \models \neg C(a) \text{ iff } a \notin C^{I(u)}$ 。

例如，在一定状态下， a 是一块积木，表明 a 是属于积木这个概念，可以表示为 $Block(a)$ ； b 是一个按钮可以表示为 $Button(b)$ 。

关系断言是用来指明在一定状态下两个个体对象之间所满足的基本关系或者某个体对象的属性，是一种二元关系。其语义解释为：

- $u \models R(a_1, a_2) \text{ iff } (a_1, a_2) \in R^{I(u)}$;
- $u \models \neg R(a_1, a_2) \text{ iff } (a_1, a_2) \notin R^{I(u)}$ 。

例如，在某状态下，主体 a_1 和主体 a_2 是熟人，可以表示为 $hasAqaintance(a_1, a_2)$ ；物体 a 压在物体 b 上，可以表示为 $On(a, b)$ 。关系断言同时也可以表示对象的一些基本属性，例如，按钮 b_1 处于开启状态，可以表示为 $hasState(b_1, ON)$ ；物体 a 的长度为 10，表示为 $hasLength(a, 10)$ 。

类似地，在一定状态 u 下，由断言公式组合而成的公式的语义可以解释如下，其中 φ, ψ 为断言公式：

- $u \models \neg \varphi \text{ iff } u \not\models \varphi$ (即在状态 u 下推导不出 φ)
- $u \models \varphi \wedge \psi \text{ iff } u \models \varphi \text{ 且 } u \models \psi$;
- $u \models \varphi \rightarrow \psi \text{ iff } u \models \varphi \Rightarrow u \models \psi$;

下面对动作的语义进行解释。动作的执行导致世界状态的变化，因此动作也可以定义为一种状态转换关系。而状态变化的过程实际上就是论域中个体的属性以及个体之间的关系的动态变化过程，因而每个状态下所有个体的属性、关系等事实的描述构成世界的状态描述，它们都可以按动作描述（定义 7）来表示。因此 DDL 中讨论的状态实际上就对应于该状态下的动作描述中对所有条件的解释，这样就可以利用上述对条件的语义解释和传统描述逻辑的语义解释方法来理解动作的语义。在定义动作的语义之前，首先定义动作如何将一个状态转换成另一个状态。

定义 2.35 给定状态 u 和状态 v 下的两个解释 $I(u) = (\Delta, \bullet^{I(u)})$ 和 $I(v) = (\Delta, \bullet^{I(v)})$ ，在状态 u 下应用动作 $\alpha = (P_\alpha, E_\alpha)$ 能产生状态 v （记为 $u \rightarrow_\alpha v$ ），如果存在一个赋值映射 $\gamma: N_X^\alpha \rightarrow \Delta$ ，使得 $\gamma, I(u)$ 和 $I(v)$ 满足下列条件：

- (1) $I(u)$ 和 γ 满足前提公式集 P_α 中的每个条件；
- (2) 对于结果公式集 E_α 中的每个序对 $head / body$ ，如果 $I(u)$ 和 γ 满足 $head$ ，则 $I(v)$ 和 γ 满足 $body$ 。

这时也称在赋值映射 γ 和状态 u 下动作 α 能产生状态 v ，记为 $u \xrightarrow[\alpha]{}^\gamma v$ 。

下面是原子动作和复杂动作的语义：

- $\alpha = \{ \langle u, v \rangle \mid u, v \in \mathcal{W}, u \xrightarrow[\alpha]{}^\gamma v \}$;
- $\alpha; \beta = \{ \langle u, v \rangle \mid u, v, w \in \mathcal{W}, u \xrightarrow[\alpha]{}^\gamma w \wedge w \xrightarrow[\beta]{}^\gamma v \}$;
- $\alpha \cup \beta = \{ \langle u, v \rangle \mid u, v \in \mathcal{W}, u \xrightarrow[\alpha]{}^\gamma v \vee u \xrightarrow[\beta]{}^\gamma v \}$;

- $\alpha^* = \{ \langle u, v \rangle \mid u, v \in \mathcal{W}, u \xrightarrow{\gamma}_{\alpha} v \vee u \xrightarrow{\gamma}_{\alpha; \alpha} v \vee u \xrightarrow{\gamma}_{\alpha; \alpha; \alpha} v \vee \dots \};$
- $\varphi? = \{ \langle u, u \rangle \mid u \in \mathcal{W}, u \models \varphi \}.$

由于上述对动作的语义解释是按传统描述逻辑的思想给出的，因而可以给出有关动作的几个新的概念定义，这些概念作为传统描述逻辑的有益补充。

定义 2.36 对于动作 α （原子动作或复杂动作），如果存在解释 $I(u) = (\Delta, \bullet^{I(u)})$ 和 $I(v) = (\Delta, \bullet^{I(v)})$ 满足 $u \xrightarrow{\alpha} v$ ，则称动作 α 是可以实现的（realizable）。

定义 2.37 对于动作 α （原子动作或复杂动作），如果存在动态描述逻辑 DDL 的有关 ABox \mathcal{A} 的解释 $I(u) = (\Delta, \bullet^{I(u)})$ 和 DDL 的一般的解释（有关 ABox \mathcal{A} 和 TBox \mathcal{T} ） $I(v) = (\Delta, \bullet^{I(v)})$ ，并且满足 $u \xrightarrow{\alpha} v$ ，则相对于 DDL 的 ABox \mathcal{A} ，称动作 α 是可以实现的。

定义 2.38 对于动作 α 和 β （原子动作或复杂动作），对任意的解释 $I(u) = (\Delta, \bullet^{I(u)})$ 和 $I(v) = (\Delta, \bullet^{I(v)})$ ，满足条件：如果有 $u \xrightarrow{\alpha} v$ ，则有 $u \xrightarrow{\beta} v$ ，则称动作 β 包含动作 α ，或者动作 α 被动作 β 包含，记作： $\alpha \sqsubseteq \beta$ 。

定义 2.39 对于动作 α 和 β （原子动作或复杂动作），动态描述逻辑 DDL 中对任意的有关 ABox \mathcal{A} 的解释 $I(u) = (\Delta, \bullet^{I(u)})$ 和一般的解释（有关 ABox \mathcal{A} 和 TBox \mathcal{T} ） $I(v) = (\Delta, \bullet^{I(v)})$ ，满足条件：如果有 $u \xrightarrow{\alpha} v$ ，则有 $u \xrightarrow{\beta} v$ ，则相对于 DDL 的 ABox \mathcal{A} ，称动作 β 包含动作 α ，或者动作 α 被动作 β 包含，记作： $\alpha \sqsubseteq_{\mathcal{A}} \beta$ 。

说明：类似定义 2.37 和定义 2.38，可以定义相对于 DDL 的 TBox \mathcal{T} 的动作 α 可实现性和包含关系。为了理解动作的包含关系，下面举一个简单的例子。

给定下列四个动作描述： $\alpha_1 = (\{A(a), \neg A(b)\}, \{\Phi/\neg A(a), \Phi/A(b)\})$ 、 $\alpha_2 = (\{A(c), \neg A(d)\}, \{\Phi/\neg A(c), \Phi/A(d)\})$ 、 $\alpha_3 = (\{A(x), \neg A(y)\}, \{\Phi/\neg A(x), \Phi/A(y)\})$ 和 $\alpha_4 = (\{A(y), \neg A(x)\}, \{\Phi/\neg A(y), \Phi/A(x)\})$ ，其中 a, b, c, d 属于个体常元， x, y 属于个体变元，则有 $\alpha_1 \sqsubseteq \alpha_3$ ， $\alpha_2 \sqsubseteq \alpha_3$ ， $\alpha_1 \sqsubseteq \alpha_4$ ， $\alpha_2 \sqsubseteq \alpha_4$ ， $\alpha_3 \sqsubseteq \alpha_4$ 和 $\alpha_4 \sqsubseteq \alpha_3$ ，但动作 α_1 和动作 α_2 之间不存在包含或被包含关系。

习 题

9. 什么是单调推理？什么是非单调推理？
10. 在缺省理论中，缺省规则是如何表示的，有哪几种表示形式？
11. 一个默认理论 $T = \langle \mathcal{W}, \mathcal{D} \rangle$ ，其中 \mathcal{D} 是默认规则集合， \mathcal{W} 是已知的或约定的事实集合。请用

默认理论表示下面的句子，并给出 D 和 W 的集合。

(1) 有些软体动物是有壳动物。

(2) 头足类动物是软体动物。

(3) 头足类动物不是有壳动物。

5. 阐述封闭世界假设和限制理论这两个非单调推理的形式化方法，并比较两者的区别？

6. 用真值维护系统描述下列情况：

(1) 现在是夏天

(2) 天气很潮湿

(3) 天气很干燥

7. 如何用真值维护系统保持知识库的一致性，并用实例说明。

8. 用情景演算描述猴子摘香蕉问题。

一个房间里，天花板上挂有一串香蕉，有一只猴子可在房间里任意活动（到处走动，推移箱子，攀登箱子等）。设房间里还有一只可被猴子移动的箱子，且猴子登上箱子时才能摘到香蕉，问猴子在某一状态下（设猴子位置为 a，箱子位置为 b，香蕉位置为 c），如何行动可摘取到香蕉。

9. 描述逻辑的基本要素是什么？

10. 动态描述逻辑如何表示动作？

第三章 约束推理

3.1 概 述

一个约束通常是指一个包含若干变量的关系表达式，用以表示这些变量所必须满足的条件。约束表示广泛地应用于人工智能的各个领域，包括定性推理、基于模型的诊断、自然语言理解、景物分析、任务调度、系统配置、科学实验规划、机械与电子设备的设计与分析等。而约束满足系统的设计是一项困难而复杂的任务，因为约束满足问题在一般情形下是一个 NP 问题，所以必须使用各种策略与启发式信息。从知识表示的角度看，也有许多重要的问题需要研究，例如表示抽象问题、默认推理问题等。非单调谓词逻辑虽然具有足够的表达能力，但它存在推理效率低、甚至不可计算的缺点，而且也不便表示启发式信息与元信息。语义网络也能表示抽象与默认信息，但本身不具备足够的问题求解能力。而带有抽象类型的约束表示可以弥补这两种表示的不足。因此，研究类型层次上的约束表示及其默认推理，是一个非常有意义的问题。

在约束推理方面，针对约束满足搜索中缩小搜索空间与控制推理代价这一对矛盾，提出了集成式的约束满足搜索算法，设计了智能回溯、约束传播及可变例示次序等策略的适当形式，并将其有机结合起来，以合理的计算代价有效地缩小了搜索空间。已有的实验结果显示该算法优于我们所知的同类算法。此外，还实现了一些特殊关系，如等式与不等式的推理，如恒等关系的单元共享策略，及不等式的图方法与区间推理的结合。这些实现将约束表达式的求值与本身的符号关系结合起来，增强了约束推理的符号演绎能力，在约束语言方面，我们设计了面向对象的约束语言 SCL，在其中实现了默认约束表示与集成式的约束推理方法；并采用常规语言中的确定型控制成分（如条件结构），而将不确定性成分局限于数据部分。从而可以使用约束传播与智能回溯来减少不确定性，缩小搜索空间。同时这种常规结构也改善了代码的可读性和语言的易学性。我们还实现了约束在 C++ 中的嵌入表示。从而使得约束程序设计充分利用 C++ 的丰富的计算资源。

一个约束满足问题 (Constraint Satisfaction Problem, 简称 CSP) 包含一组变量与一组变量间的约束。一般而言，变量表示领域参数，每个变量都有一个固定的值域。一个变量的值域可能是有限的，例如一个布尔变量的值域包含两个值；也可能是离散无限的，如整数域；也可能是连续的，如实数域。约束可用于描述领域对象的性质、相互关系、任务要求、目标等。约束满足问题的目标就是找到所有变量的一个（或多个）赋值，使所有约束都得到满足。

约束表示易于理解、编码及有效实现，它具有以下优点：

(1) 约束表示允许以说明性的方式来表达领域知识，表达能力较强，应用程序只需指定问题的目标条件及数据间的相互关系。因而具有逻辑表示的类似性质。

(2) 约束表示允许变量的域包含任意多个值，而不像命题只取真假二值。所以它保存了问题的一些结构信息，如变量域的大小、变量间的相关性等，从而为问题求解提供启发式信息。

(3) 易于并行实现。因为约束网络上的信息传播可以认为是同时的。

(4) 适合于递增型系统。约束可以递增式地加入到约束网络。

(5) 易于与领域相关的问题求解模型相衔接。各种数学规划技术，方程求解技术等，都可以自然地嵌入约束系统。

经过多年研究，人们提出了不少约束推理的方法。根据联系于约束网络节点上的数据类型，可以将约束推理分为以下几种。

(1) 关系推理：推理过程中推出的新的约束关系，并将其加到约束网络中。Kuiper 的 ENV 系统、Simmon 的 Quantity Lattice 系统，及 Brooks 的 CMS 系统，都属于关系推理。

(2) 标记推理：每个节点标注以可能值的集合，在传播过程中约束用于限制这些集合。

(3) 值推理：节点标记以常量值。约束用已标记节点的值求出标记节点的值。SKETCHPAD 及 THINGLAB 都使用值推理。

(4) 表达式推理：是值推理的推广，其中节点可能标以关于其它节点的表达式。当一个节点标记以不同的表达式时，应使其等同起来，并求解结果方程。CONSTRAINTS 就使用这种推理。

约束变量的取值可能是数值，也可能是非数值，即符号值。一般而言，非数值变量的取值范围是一个有限集合。因而当约束传播停止后，总可以进行穷举搜索来确定其一致性。而数值变量则不然。数值变量通常有无限值域，不可能进行穷尽搜索。

上述几种约束推理都有某些不足。例如，值推理只能用于方程约束，而不能用于不等式约束。关系推理与标记推理难以控制，且很难防止其进入无限循环。在关系推理中，难以确定新推出的约束是否对给定的问题有用。但标记推理要好一些，可用于任意形式的约束。

现有的约束表示可分为几类，按复杂性的次序列举如下：

- 一元谓词。
- 序关系语言，只包含偏序关系或实变量上的大小关系。
- 形如 $x - y > c$ 或 $x - y \geq c$ 的方程。
- 单位系数的线性方程与不等式，即所有的系数为 -1, 0, 1。
- 任意系数的线性方程与不等式。
- 约束的布尔组合。
- 代数与三角方程。

最简单形式的约束是一元谓词，即对变量的标记，几种最重要的标记是符号、区间、与实际值。

序关系出现在只关心数量的大小关系的系统中。例如有些系统只关心事件的次序而不考虑其时间区间，如 NOAH 系统。在 NOAH 系统中，规划的每一级指定了该级中动作的偏序关系。

形如 $x - y \geq c$ 的不等式在只知道变量之间的差的系统中非常有用。这种约束在 TMM 及

许多任务规划程序得到广泛应用。

在具有标量乘法的度量空间中，对两个数量的商的界定有时也是非常有用的。不过这种表示与差的界定是同构的。同构映射为对数函数。Allen 与 Kantz 使用商界定与序关系的布尔组合来实现时态推理。

单位系数的线性方程在常识推理是很有用的。因为这种关系对定性地表示守恒法则是足够的。守恒法则断言一个值的变化等于其增加的总和减去其减少的总和。如果我们只对状态变化感兴趣，则变量可仅取 +, -, 0 三个定性值，其系数总可为单位系数。

线性不等式是应用十分广泛的一种约束。约束的布尔组合在物理推理、电路设计及规划中也得到广泛应用。在物理推理中还经常涉及非线性方程与不等式。在几何推理中，还经常涉及代数与三角方程。

目前，约束推理的研究主要集中于两个方面：约束搜索与约束语言。约束搜索主要研究有限域上的约束满足。对有限域而言，约束满足问题一般情况下是一个 NP 问题。目前大体包括下列方法：

- 回溯法。
- 约束传播。
- 智能回溯与真值维护。
- 可变次序例示。
- 局部修正法。

约束推理研究的另一个主要方面是约束语言。以下是几个比较典型的约束语言：

(1) CONSTRAINTS

CONSTRAINTS 是一个面向电路描述的约束表示语言。作为一个约束表示语言，它使用了符号处理技术来求解数学方程。在 CONSTRAINTS 中，物理部件的功能及器件的结构都用约束表示。这些约束一般是线性方程与不等式，也包括条件表达式。约束变量一般是表示物理量的实变量。也有一些取离散值的变量。如开关的状态、三极管的工作状态等。系统采用表达式推理与值推理。并实现相关制导的回溯。

CONSTRAINTS 的一个优点是在类型层次中表示约束，用约束来表示物理对象的功能与结构。其缺点是该语言缺乏类似于面向对象语言中的方法那样的成分，不能定义特定于某个类的概念。同时，约束传播方法比较单一，既缺乏实域上的区间传播机制，也缺乏有限域上的域传播机制。

(2) Bertrand

Bertrand 是由 Leler 开发的一个高级约束语言。它的计算模型是基于增强型项重写技术的系统。基本上是在项重写系统的基础上加上赋值功能与类型机制。这种技术使得 Bertrand 能够解决实数与有理数上的线性方程。Bertrand 还包括抽象数据类型的功能。它已被用来解决图形学及电路上的一些例子。

Bertrand 还不能算是真正的约束语言，而只能算是约束满足系统的生成工具。它基本

上没有提供什么求解离散约束的机制，也没有提供实数域上的区间传播或序关系传播的机制，而只能进行值传播。

(3) 约束逻辑程序设计语言 CHIP

约束语言很大一部分研究集中于约束逻辑程序设计语言。其宗旨在于将约束满足技术与逻辑程序设计结合起来，基本上是在 Prolog 的基础上引入约束传播机制（主要是弧一致性技术），以提高搜索效率，增强表达能力。CHIP (Constraint handling in Prolog) 就是这样较有影响的一个约束逻辑程序设计语言，其目的是简便、灵活而有效地解决一大类组合问题。它通过提供几种新的计算域而增强逻辑程序设计的能力；有限域、布尔项及有理项，对于每个计算域，都提供有效的约束求解技术，即有限域上的一致性技术，布尔域的布尔合一技术及有理数域上的单纯型法。除此以外，CHIP 还包含一个一般的延迟计算机制。

CHIP 主要应用于两个领域：运筹学与硬件设计。CHIP 缺乏类型机制，而这种机制对于表达领域概念是极其重要的。

(4) 约束层次与 HCLP

约束满足研究中一个非常有趣而且有着重要实用价值的问题是所谓“软约束”的求解问题。其中比较有影响的是 Borning 等人的工作 [Freeman-Benson 1992]。Borning 为了解决图形界面设计问题中“限制过度” (overly constrained) 的问题，提出了约束层次的概念。其基本思想是除了必须满足的“硬约束”之外，还允许用户表达需要尽可能满足的“软约束”。这些软约束被分成若干优先等级。这种表示就称为约束层次。

约束层次是带标记的约束的有限集合。给定一个约束层次 H ， H_0 是 H 中必须成立的约束的集合。 H_1 是在非必要约束中最强的约束，如此等等。最弱的约束为 H_n 。 n 是约束层次中非必要约束的级数。

对一个约束集合的求值是一个函数。它将自由变量映射到域 D 中的元素。约束层次的一个解是自由变量的一个求值的集合。解集中的求值必须至少满足必要约束。除此以外，这个解集还至少满足其它求值所满足的约束。即在满足必要约束的求值中，不存在比解集求值满足更多非必要约束的求值。有许多种比较求值对约束的满足程度的方法。具体采用哪一种更好则取决于应用。

最初的约束层次的求解是用一些过程（方法）来使约束层次得以最大程度地满足。但这种过程表示丧失了约束表示系统所应有的说明性优点。因而 Wilson, Borning 等人在约束逻辑程序设计的基础上，设计了层次型约束逻辑程序设计语言 HCLP。该语言将约束层次嵌入 Prolog 之中，使整个约束层次系统完全建立在说明性表示基础之上。他们用非单调理论中的最小模型理论给出了 HCLP 的模型论语义。在实现上，首先用普通的逻辑程序的回溯方法求得一个必要解。这个解可能包含对某些变量的域（如区间）的限定，而不是变量的具体值。同时还生成当前例示下的所有约束（即约束层次），然后由较强的约束到较弱的约束，逐次用所生成的约束去限定所生成的必要解。直到出现不一致为止。对于相同层次中的约束的不同的使用次序，将导致不同的解。

约束层次的这种解法是简单而快捷的，但却是不完全的，因为由此而产生的解在一般情况下并不一定对应整个逻辑程序的最小模型，而只对应一个推理路径中的最小模型。

(5) 面向对象约束语言 COPS

中国科学院计算技术研究所智能计算机科学开放实验室研制的 COPS 系统利用面向对象技术，将说明性约束表达与类型层次结合起来。在形式上吸收了常规语言，主要是面向对象的程序设计语言的基本形式。内部求解时采用约束推理机制，使说明性约束表达式与类型层次相结合，实现知识的结构化封装，充分发挥两者的优点，力图实现一个具有较强表达能力和较高求解效率的约束满足系统。COPS 的设计考虑了软件工程的应用要求，尽量将一个不确定问题确定化：它允许条件语句与循环语句，而不是单纯以递归的形式来实现迭代计算；通过类方法的重载实现同一约束的不同实现，提高了程序的执行效率。COPS 系统同时是一个渐增式的开放系统，用户能通过类型层次定义，实现新的数据类型和新的约束关系。约束语言 COPS 具有许多人工智能程序设计语言的特点，如约束传播、面向目标和数据驱动的问题求解、有限步的回溯、对象分层中的继承等。

(6) ILOG

ILOG 公司创建于 1987 年，总部位于法国巴黎和美国加州，是全球领先的优化、互动图像界面以及商业规则应用领域的软件组件供应商，也是全球将优化算法运用到商业应用软件中的公司。产品应用遍布于电信，交通，国防，电力，物流等领域。ILOG 公司的 ILOG Solver 使用建模语言来表示约束问题。

3.2 回溯法

求解有限约束满足问题的最简单直接的方法是生成测试法，即依次生成所有变量的值的各种组合，对其进行测试，直到一个测试成功的组合。这种方法显然是低效的。一种直接的改进是顺序回溯法。顺序回溯法以固定次序对变量进行例示，当新的变量与先前赋值的变量不一致时，它尝试其变量域中的其它值，直到域中所有的值都被穷尽。当所有的值都失败，则回到上一个赋值的变量，并对该变量重新赋值。

假设一个 CSP 问题的解由一个不确定长度的向量组成，即 (x_1, x_2, \dots) 它满足问题中所有约束条件。设变量 x_i 的值域为 X_i ，则 x_i 的取值只能是 X_i 中的某个元素，而问题的整个可行解空间为

$$X_1 \times X_2 \times \dots \times X_n$$

其中 n 是变量总数。

回溯算法求解开始时，部分解为空向量。然后从变量集合中选择一个变量 x_i 并加到部分解中去。通常可选一个最小的元素作为 x_i 的取值。各种约束会告诉我们 X_i 中哪些成员可作为 x_i 的候选者，这些成员可用一个子集 S_i 来表示。由约束条件可以找出从部分解 $(x_1, x_2, \dots, x_{k-1})$ 到部分解 $(x_1, x_2, \dots, x_{k-1}, x_k)$ 的候选者。如果 $(x_1, x_2, \dots, x_{k-1})$ 不允许 x_k 取任何值，那么 $S_k = \emptyset$ ，必须进行回溯，并为 x_{k-1} 选取一个新的允许值。如果 x_{k-1} 没有新的允许值，就进一步回溯至 x_{k-2} ，以此类推。设 $T(x_1, x_2, \dots, x_{k-1})$ 表示 x_k 的所有可能取值。当部分解 (x_1, x_2, \dots, x_k) 不允许有新的扩展结点，则限界函数 $BT_k(x_1, x_2, \dots, x_k)$ 取假，否则取真值。只求一个解的回溯算法如

下:

算法 3.1 回溯算法 BACKTRACK。

输入: 一个 CSP 问题

输出: 一个完全解或无解返回

```
procedure BACKTRACK
begin
    k=1;
    while k>0 do
        if  $x_k$  存在未检验过的值使
             $x_k \in T(x_1, x_2, \dots, x_{k-1})$  and
             $BT_k(x_1, x_2, \dots, x_k) = \text{true}$ 
        then if  $(x_1, x_2, \dots, x_k)$  满足所有约束条件
            then return(0); /* 返回一个解 */
            else k=k+1;
        endif;
        else k=k-1;
    endwhile
    return(1); /* 无解返回 */
end BACKTRACK
```

尽管回溯法好于生成测试法,但对于非平凡问题仍然是低效的。其原因在于搜索空间中不同路径的搜索重复相同的失败子路径。一些研究者认为,造成这种反复的原因是所谓的局部不一致性。最简单的情形是所谓的结点不一致性。对一个变量 v_i 的一个一元约束。存在域中一个值 v_i 不满足该约束。这样,每当 v_i 取到 a 时就会出现不一致性。另一种重复的情形是所谓的弧不一致性。可以用下例来说明。设变量的例示次序为 $v_1, v_2, \dots, v_i, \dots, v_j, \dots, v_n$ 。设 v_i 与 v_j 之间存在约束,使得对 $v_i=a$, v_j 不存在任何值满足该约束。这样每当 v_i 赋以 a 值后,当 v_j 赋值时将出现失败。这个失败将对所有 $v_r (i < r < j)$ 的所有值的组合都重复。

因为节点不一致性而出现的反复,可以通过消除每个变量域中不满足其一元约束的值而加以克服。而弧不一致性可以通过执行弧一致性算法而加以消除。下一节给出弧一致性(算法)的形式定义并给出相应的算法。

3.3 约束传播

如果对 v_i 的当前域中的所有值 x , 存在 v_j 的当前域中的某值 y 使得 $v_i=x$ 和 $v_j=y$ 是 v_i 与 v_j 之间的约束所允许的, 则弧 (v_i, v_j) 是弧一致的。弧一致性的概念是有向的。即 (v_i, v_j) 是弧一致的并不自动地意味着 (v_j, v_i) 是一致的。例如, 如果 v_1 的当前域为 $\{a\}$, v_2 的当前域为 $\{a, b\}$, v_1 与 v_2 之间的约束为不等关系, 则 (v_1, v_2) 是弧一致的。而 (v_2, v_1) 则不是。因为对 $v_2=b$, 不存在 v_1 域中任何值, 使其不等关系成立。显然 (v_i, v_j) 之间可以通过删除不满足彼此之间不满足约束的值而实现其弧一致性。而且删除这些值不影响原来 CSP 的任何解。以下就是实现这种删除操作的算法。

算法 3.2 约束传播修改算法[Mackworth 1977]。

```
procedure REVISE( $V_i, V_j$ )
DELETE  $\leftarrow$  false;
```



```

for each  $x \in D_i$  do
  if there is no such  $y \in D_j$ 
    such that  $(x, y_j)$  is consistent,
  then
    delete  $x$  from  $D_i$ ;
    DELETE  $\leftarrow$  true;
  endif
endfor
return DELETE;
end REVISE

```

为了使约束网络中所有的弧都一致，只对每个弧执行一次 REVISE 操作是不够的。每当 REVISE 对某个变量 v_i 进行了删减，以前修改过的弧 (v_i, v_j) 必须重新修改，因为 v_i 域已变小了。以下算法对整个约束网络获得弧一致性。

算法 3.3 约束传播 AC-1 算法[Mackworth 1977]。

```

procedure AC-1
 $Q \leftarrow \{(V_i, V_j) \in \text{arcs}(G), i \neq j\}$ ;
repeat
  CHANGE  $\leftarrow$  false;
  for each  $(V_i, V_j) \in Q$  do
    CHANGE  $\leftarrow$  REVISE( $V_i, V_j$ )  $\vee$  CHANGE;
  endfor;
until not(CHANGE);
end AC-1

```

这个算法的主要缺点是，一次成功的修改后，对所有的弧都要作一次修改操作。尽管仅有少数弧受到影响。AC-3 对此作了改进，该算法仅对可能受到影响的弧进行修改操作。

算法 3.4 约束传播 AC-3 算法[Mackworth 1977]。

```

procedure AC-3
 $Q \leftarrow \{(V_i, V_j) \in \text{arcs}(G), i \neq j\}$ ;
while  $Q$  not empty
  select and delete any arc  $(V_k, V_m)$  from  $Q$ ;
  if (REVISE( $V_k, V_m$ ))
    then  $Q \leftarrow \{(V_i, V_k) \text{ such that } (V_i, V_k) \in \text{arcs}(G), i \neq k, i \neq m\}$ 
  endif;
endwhile;
end AC-3

```

著名的 Waltz 算法是这个算法的特例[Waltz 1975]。它等价于 Mackworth 提出的另一个算法 AC-2。假定每个变量域的大小为 d ，约束网络中边的个数为 e ，弧一致性算法的复杂性为 $O(ed^3)$ 。Mohr 与 Henderson 提出了另一个弧一致算法[Mohr 1986]，其复杂性为 $O(ed^2)$ 。因而就最坏情形的复杂性而言，Mohr 与 Henderson 的算法是最优的。

给出一个弧一致的约束网络，是否存在变量当前域中的一个赋值。使其成为 CSP 的一个解。如果每个变量的域最后都只剩下一个值，答案无疑是肯定的。否则在一般情况下答案是

否定的。尽管如此，弧一致性算法还是减小了回溯法的搜索空间。

既然弧一致性算法不足以代替回溯法，那么是否存在更强的一致性算法能够消除对回溯法搜索的需要呢。 k 一致性的概念对不同的 k 值体现不同的一致性。一个约束网络是弧一致的而且仅当对任意 $k-1$ 个变量的值，如果这些值满足这些变量之间的所有约束，则对任意第 k 变量存在一个域中的值，它与 $k-1$ 个值一起满足这 k 变量的所有约束。更精确地说：

设变量 X_1, X_2, \dots, X_{k-1} 的赋值分别为 a_1, a_2, \dots, a_{k-1} 。如果 a_1, a_2, \dots, a_{k-1} 满足 $V_1, V_2, \dots, V_{k-1}, V_k$ 间的所有约束，则对任意第 k 个变量 V_k ，存在一个值 a_k ，使得 $a_1, a_2, \dots, a_{k-1}, a_k$ 满足 $V_1, V_2, \dots, V_{k-1}, V_k$ 间的所有约束。

一个约束网络是强 K -一致的，如果对所有的 $J \leq K$ ，都是 J -一致的，节点一致性相当于1强一致的，弧一致性相当于2强一致的。对 $K > 2$ ，存在算法使得约束网络是强一致的 (Cooper 1989)。虽然，如果一个有 n 个节点的约束网络是 n 强一致的，则不用搜索就能找到一个解。然而对 n 节点约束网络，其 n 一致算法也是指数的。

不过，对于具有特殊结构的约束网络，存在多项式算法。最简单的情形是，对于树结构的约束网络存在弧一致算法，使其可以在线性时间内求解。

3.4 约束传播在树搜索中的作用

上面讨论了两种不同的 CSP 求解方法：回溯法与约束传播。在第一种方法中通过测试变量的不同组合来得到一个完整的解。这种方法的缺陷是搜索路径的重复。在第二种方法中，通过传播变量间的约束，从而降低问题的复杂性。尽管任意 CSP 问题总可以通过 n 一致性算法来求解。但 n 一致性算法比回溯法效率更低。而对 $k < n$ ， k 一致性算法又不能保证能得到一个全局一致的解。一种综合的办法是将给传播嵌入到回溯法中，具体做法如下。

首先生成一个根搜索节点，求解最初的 CSP。当一个搜索节点被访问，使用约束传播算法以达到预期程度的一致性。如果存在一个搜索节点，每个变量恰包含一个值，而且对应的 CSP 是弧一致的，则该搜索节点表示一个解。如果在约束传播过程中，某些变量的域变为空，则该搜索节点被剪枝。否则，选择一个变量（其域的大小 > 1 ），对该变量的每一个可能的值生成新的 CSP 搜索节点。每个新的 CSP 搜索节点表示当前 CSP 搜索节点的后继节点。所有这些搜索节点通过使用深度优先的回溯法来访问。

现在问题是对每个搜索节点约束传播应做到何种程度。如果完全不做约束传播，则退化成本标准的回溯法。如果对具有 m 个未赋值的变量的 CSP 搜索节点施行 m -一致性算法则等于完全排除了回溯法的使用而造成严重的低效。经验表明，受限形式的约束传播（一般其一致性程度不超过弧一致性）具有最佳的效益。

3.5 智能回溯与真值维护

克服标准回溯法缺陷的另一种方法是智能回溯。在智能回溯中，回溯到哪一步决于是哪一个变量造成了失败。智能回溯是如何克服标准回溯的缺陷，可以从如下例子来考虑。

假定变量 v_1, v_2 与 v_3 已分别赋值 a_1, b_3, c_2 。假定 v_3 中尚未发现有任何值与 v_2 的 b_1 和 b_2 相容。现在假定 v_4 的所有可能值都与 $v_1=a$ 相矛盾。因为矛盾是由 v_1 的不适当选择所引起的，所以智能回溯应回溯到 v_1 。但这种方法也同时撤消 v_2 与 v_3 的值。

这种方法虽然能够根据失败的源找到正确的回溯点，但并没有完全避免相同路径的重复。相关制导的回溯是解决这个问题的一种方法。这种方法在 Doyle 的真值维护系统 (TMS) 中

得到推广与应用。

一个基于真值维护系统的问题求解系统包含两个部分：一个推理机与一个真值维护系统 TMS。推理机用于由旧的事实推出新的事实，而真值维护系统 TMS 记录这种推理的论据。新事实的加入可能使得某些已有的假设不再为真。因而这种论据的维护使得可以废除不再成立的假设。这种方法对 CSP 的应用可描述如下。

当一个变量被赋以某个值，生成该赋值的一个论据。类似地，对一个默认赋值也可生成其论据。对这种情况，系统检查当前赋值是否违反任何约束。如果违反，则生成一个新的节点表示两个变量的一对值相互矛盾。这种节点还被用来作为其它赋值的论据，这个过程一直持续到对所有变量找到一个一致的赋值。这种系统不会出现冗余与重复的计算。

尽管这种系统的搜索量是最小的，但确定违反约束的失败源的开销却是极大的，由于推理步数是指数增长的，因而用于存储及检索的时间与空间开销也都是指数复杂的。所以对很多问题，这种方法可能比普通回溯法占用更多的时间。

另一个相关的工作是 de Kleer 的基于假设的真值维护系统 ATMS。基于假设的真值维护系统由真值维护系统 TMS 演变而来。真值维护系统可视为某种形式的约束传播。ATMS 是 TMS 的扩充，TMS 只维持单个上下文，而 ATMS 则试图同时搜索多个上下文。因而，每个推出的事实都与所有使其成立的假设相联系。一个结论可能在多个上下文都成立。假设与论据的区别在于，除非有证据其为假，否则总认为是真。ATMS 的优点在于不同的上下文可以共享一些中间假设，而不用去分别地生成它们。但中间假设的个数通常是指数增长的，因而这种生成过程是呈指数复杂性的。但 ATMS 具有如下一些缺点：

- 它们是面向求全解的，对求单解经常出现不必要的搜索；
- 用于维护所有上下文的时间与空间开销很大；
- 排错比较困难。

而且这种方法很难估计其存储要求，所有的推导及其假设都必须记录。这使得这种方法很难应用到大规模系统上，de Kleer 与 Williams 后来又建议回到回溯的方法以控制 ATMS。这也就类似于一种智能回溯方法了。

3.6 变量例示次序与赋值次序

对标准回溯法的另一个可以改进的方面是变量的例示次序。实验表明这种次序对回溯搜索的效率有着巨大的影响。已提出多种启发式策略。一种是选择当前域最小的变量最先例示。这样，变量的例示次序一般是动态确定的，而且对于搜索树中的不同分支可能是不同的。Purdon 与 Brown 广泛地研究了这种启发式及其变型[Purdon 1983]，他们的结果表明对标准回溯法有着极大的改进。

另一个启发式是优先例示参与约束最多的变量。这种方法的出发点是不成功的分支尽可能早地剪除。

前面提到过树结构约束网络可以不回溯，而对于求解任何一个约束网络可以通过删除某些节点而使其成为树，这些节点的集合称圈割集。如果能找到一个小的圈割集，则一个好的启发式是先例示割集中的变量，然后求解剩下的树结构 CSP。

3.7 局部修正搜索法

目前绝大部分算法是基于树搜索的构造性求解方法。但近来另一种求解组合问题的方法

以其惊人的实验结果引起了人们的注意，这就是所谓的“局部修正法”[Gu 1992]。这种方法首先生成一个全部的，但可能是不一致的变量赋值，然后针对所出现的矛盾改变某个变量的值，以减少所违反约束的个数，如此反复，直到达到一个一致的赋值。该方法遵循一个简单的法则：找到一个引起矛盾的变量，然后选一个新值赋给它，使得结果赋值将矛盾减到最少。该方法的基本思想如下。

给定：一个变量的集合，一个二元约束的集合，及对每个变量的一个赋值。两个变量相矛盾如果它们的值违反了一个约束。

过程：选一个矛盾变量，并为它赋一个值使得将矛盾的个数减到最小。

局部修正搜索法十分有效的主要原因在于，对每个变量的赋值提出了较多的信息，因而下一个所转换的状态较大幅度地减少了矛盾的个数。

局部修正搜索法也有其局限性。首先这种策略是不完备的。当问题没有解时，该过程可能不终止。当问题的解密度比较小时，搜索的效率也很低。我们曾对图着色问题进行过修正法与树搜索之间的比较。在图的边较少时，修正法较优。反之树搜索效率高。如果问题无解，修正法不终止。

目前，约束搜索的一个新的动向是研究 CSP 的难度分布。尽管 CSP 一般说来是一个 NP 问题，但大量的 CSP 可以在可容忍的时间内解决。真正难解的问题只是一小部分。

3.8 基于图的回跳法

基于图的回跳法(Graph-based Backjumping)属于相关制导的回溯中的一种。其中的相关信息来自约束网络的图结构。一个约束图是指由约束关系所隐含的图。其中的顶点为变量，边为约束（这里假定所有的约束都是二元的）。即两个顶点有一条边仅当它们所代表的变量存在约束。在基于图的回跳法中，每当在某个变量 x 出现失败，算法总是回到在图中与 x 联结的最近赋值变量。下面是基于图的回跳法的算法。

算法 3.5 前向传播算法 Forward。

```
Forward( $x_1, \dots, x_i, P$ )
begin
  if  $i=n$  then 退出并返回当前赋值;
   $C_{i+1} \leftarrow \text{computeCandidates}(x_1, \dots, x_i, x_{i+1})$ ;
  if  $C_{i+1}$  非空, then
     $x_{i+1} \leftarrow C_{i+1}$  中的第一个元素;
     $C_{i+1}$  中删除  $x_{i+1}$ ;
    Forward( $x_1, \dots, x_i, x_{i+1}, P$ );
  else
    Jumpback( $x_1, \dots, x_i, P$ );
  endif
end
```

其中 P 用以保留将回溯的变量的集合。

算法 3.6 回跳算法 Jumpback。

```
Jumpback( $x_1, \dots, x_i, x_{i+1}, P$ )
begin
```

```

if  $i=0$ , then 退出无解;
PARENTS  $\leftarrow$  Parents( $x_{i+1}$ );
 $P \leftarrow P \cup$  PARENTS;
设  $j$  是  $P$  中编号最大的变量,  $P \leftarrow P - x_j$ ;
if  $C_j \neq \emptyset$ , then
     $x_j = \text{first in } C_j$ ;
    从  $C_j$  中删除  $x_j$ ;
    Forward ( $x_1, \dots, x_j, P$ );
else
    Jumpback ( $x_1, \dots, x_{i-1}, x_j, P$ );
endif;
end

```

3.9 基于影响的回跳法

对基于图的回跳法的改进，形成新的基于影响的回跳法。首先，我们的目标是利用动态相关信息，而不仅仅是静态联结关系。这里问题的关键在于如何将引入动态信息所造成的计算资源的开销限定在合理的范围内。通过对算法及其数据结构的精巧设计，我们成功地解决了这一问题。

基于影响的回跳法 IBMD (Influence-based-Backjumping, Most-constrained-first, Domain-filtering) 将三种策略综合在一起：最约束者优先 (Most-Constrained-First)、域筛减法 (Domain Filtering)、回跳法。最约束者优先是指每次选“自由度”最小的变量加以赋值。这里的“自由度”主要用变量当前取值域的大小加以衡量，因为它是最重要而又最容易计算的一个量。也可辅之以该变量在约束网络中的关联度。最约束者优先策略是对固定次序赋值策略的一项十分有效的改进。

域筛减法就是从变量域中删除与已赋值的变量不一致的值。这是一种代价较小的约束传播技术。每当一个新的变量赋值时，未赋值变量中与该赋值不一致的所有值都被从域中删除。但因为这种赋值是试探性的，所以被删除的值必须保存起来。当某个赋值被撤回时，由该赋值所引起的所有筛除都必须恢复。

我们所设计的回跳法与传统的依赖制导的回溯，包括基于图的回跳法，有较大区别。传统的依赖制导的回溯显式地记载一个推论所依据的假设集合。每当回溯一步时，都要把被撤回的变量所依赖的假设集合合并到该集合中最近赋值的变量的假设集中。这种归并造成了很大的时间与空间的开销。为了避免这种开销，我们的回跳法并不是根据所记录的依赖假设集来回跳（事实上我们根本不需要这种记录）。我们是根据筛减过程中赋值变量对赋值变量的实际影响来回跳。一个赋值对一个变量 v_i 产生影响，当且仅当 v_i 当前域中至少有一个值被该赋值所引起的筛减过程删除。显然这种影响关系比网络的联结关系更精确。因为联结关系这是一种可能的数据相关性，而影响关系则是实际的数据依赖性。因而，可能某两个变量在约束图中是相连的，但在当前上下文下并未发生影响关系。基于影响的回跳法的空间规模要比基于图的回跳法来得小。由此，还产生时间开销的减少。

在 IBMD 中，每个例示变量都保存一个受影响的变量的集合。给定 IBMD 的一个上下文（所有例示变量的赋值），一个变量 v_i 受一个例示变量 v_j 的影响当且仅当 v_j 的例示使 v_i 的当前域有所减小。

IBMD 包括前向例示过程和反向回溯过程。前向例示过程首先选择具有最小当前域的变量赋值。对所选变量 v_i 赋值后, 执行域筛减过程。对所有未赋值的变量 v_j , 删去其值域 D_j 中与 v_i 的赋值相矛盾的值。如果 D_j 变为空, 则意味着 v_i 的赋值与以前的变量赋值相矛盾, 因而开始回溯。回溯过程首先将所有因 v_i 的例示而删去的变量的值恢复到该变量的域中。如果 v_i 又穷尽了所有的值, 则继续回溯上一个例示变量 v_h 。如果 v_h 对 v_j 或任一已穷尽的变量没有任何影响, 或 v_h 已穷尽了所有的选择, 则令 v_h 为再上一个例示变量, 重复该回溯过程, 设该过程停止在 v_g , 且影响 v_g 的最新变量为 v_f , 则将 v_g 的例示值从 D_g 中删除, 将 v_g 及其例示值记入 v_f 的影响集中。然后重新对 v_g 赋值, 从而回到前向例示过程。下面是算法的 C 语言程序。

算法 3.7 基于影响的回跳算法 IBMD[廖乐健 1994b]。

```
IBMD()
{
    int var, failvar;
    while (uninstantiated != nil ) {
        var=mostConstrained();
        /* Choose a variable to instantiate */
        a[var]=domain[var][0]; /* Assign a value. */
        while ((failvar=propagate(var))!=SUCCESS) {
            /* While inconsistent */
            if ((var=backjump(failvar)==nil) return(0);
            /* If back up to top, exit */
            a[var]=domain[var][0]; /* Else reinstantiate var */
        }
    }
    return(1);
}
```

算法 3.8 回跳算法 backjup(廖乐健 1994b)。

```
backjump (failvar)
int failvar;
{
    int assignedVar, sourceVar;
    failedVarSet=makeVariableSet();
    addVariableSet(failvar, failedVarSet);
    /* Initialize the set of failed variables */
    while ( (assignedVar=lastAssigned[ASSIGNEDHEAD]) !=TOP)
    { /* Until go back to top */
        retract(assignedVar); /* Retract the influences of the variable */
        if (relevant) /* The instantiation influence failedVarSet */
            if (! exhausted(assignedVar)) break;
            else addVariableSet(assignedVar, failedVarSet);
    }
    if (assignedVar==TOP) return(nil);
    for(sourceVar=lastAssigned[assignedVar];
```

```

        sourceVar !=TOP && ! isCulprit(sourceVar);
        sourceVar=lastAssigned[sourceVar]);
        /* Find last variable that is relevant to the failure */
        domainfilter(sourceVar, assignedVar, a[assignedVar]);
        /* Filter the current value and record the influence */
        return(assignedVar);
    }

```

在程序中，函数 mostConstrained () 选择一个当前域最小的变量，并将其从未赋值的变量表 uninstantiated 移到已赋值的变量表 instantiated 中。propagate() 完成域筛减过程。如果某变量筛减后的域为空，则返回该变量，否则返回 0。函数 retract() 执行回跳过程，并返回回跳所落到的变量。如果回跳到顶，则返回 nil 值。数组 a[] 用来存放各个变量所赋值的值。二维数组 domain[][] 用来对每个变量的每个值记载该值的状态：属于当前域，或者被某个前面所例示的变量所删除。属于当前域的值用下标链在一起。属于由同一个变量所删除的值的集合也用下标链在一起。

该算法由于实施穷尽搜索，因而具有指数时间复杂性。该算法的空间复杂性为 $O(n(n+d))$ ，其中 n 为变量个数， d 为变量域的大小（假设所有域具有相同大小）。

用 IBMD 测试了一系列的例子，并与其它一些算法作了比较。这些算法包括顺序回溯法 (BT)、基于冲突的回跳法 CBJ、仅采用最约束者优先的算法 MCF、仅采用最约束者优先与域筛减法的算法 MD。试验结果显示 IBMD 在不同程度上优于这些算法。例如，在 工作站 sparc-1 上，用 IBMD 运行 N 皇后的例子。IBMD 对 $N=100$ 用了不到 2 秒钟。而在相同的机器上，用 BT 算法运行 20 皇后问题用了 10 多秒钟。当 $N \geq 30$ 时，BT 算法至少需要若干小时。对 MCF 算法，当 $N=50$ 时需用 10 多秒钟。当 $N \geq 60$ 时需要若干小时。对 N 皇后问题，MD 算法与 IBMD 算法的性能大体相当。IBMD 略优于 MD。这里有几种解释。首先，在 N 皇后问题中，超过一步的回跳很少。域筛减策略消去了许多“表层”不一致性，而“深层”不一致性涉及许多其它变量。其次 IBMD 的额外开销较小。表 3.1 给出了运行 N 皇后问题各种算法的执行时间。

我们还试验了图作着色的例子。当结点的联结点度较小 (≤ 7) 时，几乎不需要回溯。此时 IBMD 和 MD 性能相当。当增加联结点度时，IBMD 明显优于 MD。

下面将算法 IBMD 与同类工作的其它算法进行比较。

(1) 与最初的回跳法 BJ 相比。

BJ 只能由出现不一致的变量回跳到造成其不一致性的最新变量。但如果该变量已穷尽了所有的选择，再往下的回溯与普通的顺序回溯法相同。因而这种方法对性能的改进很有限。而 IBMD 则可以连续往下回溯，直到找到一个与不一致性相关且尚未穷尽所有值的变量。

(2) 与基于图的回跳法相比。

约束网络的拓扑联结性是数据实相关性的近似。在特定上下文下，很可能两个图相关的变量并不发生影响。而 IBMD 利用的是实际的影响关系。

(3) 与 Prosser 等人最近的工作相比。

Prosser 等人在 93 年 IJCAI 上发表了对回跳法的一些改进。提出了基于冲突的回跳法 CBJ (Conflict-based BackJumping)。IBMD 与 CBJ 的出发点是类似的，但在实现上有很大不同。CBJ 存在一个缺点。每一个变量都有一个依赖变量集。每当回跳一次时，都要执行依赖集的合并操作，因而造成了很大的时间开销。而 IBMD 不存在这个问题。

(4) IBMD 优于所有上述算法的一个方面是将最约束者优先及域筛减策略结合进来。

表 3.1 运行 N 皇后问题的执行时间（单位秒）

n	BT	CBJ	IBMD	n	BT	CBJ	IBMD
8	0.0	0.0	0.0	26	324.0	323.6	0.0
9	0.0	0.0	0.0	27	389.4	389.5	0.0
10	0.0	0.0	0.0	28	2810.2	2799.0	0.0
11	0.0	0.0	0.0	29	1511.1	1517.2	0.0
12	0.1	0.0	0.0	30	*	*	0.1
13	0.0	0.1	0.0	40	*	*	0.3
14	0.5	0.5	0.0	50	*	*	1.8
15	0.4	0.5	0.0	60	*	*	0.7
16	3.3	3.7	0.0	70	*	*	1.4
17	1.9	2.2	0.0	80	*	*	1.0
18	16.9	17.9	0.0	90	*	*	1155.7
19	1.1	1.3	0.0	100	*	*	1.8
20	101.6	105.5	0.0	110	*	*	158.9
21	4.6	4.7	0.0	120	*	*	*
22	1068.1	1048.4	0.0	130	*	*	11.5
23	16.2	16.2	0.0	140	*	*	*
24	290.5	289.6	0.0	160	*	*	7.1
25	35.8	35.9	0.0	200	*	*	14.0

注：*表示大于 2 小时

3.10 约束关系运算的处理

3.10.1 恒等关系的单元共享策略

在约束推理中，如果两个变量总是取相同的值，这两个变量是恒等关系。恒等关系是一类重要的约束关系。例如，电路中两个部件的串联关系，意味着流过两个部件的电流相等。一个正方形可以定义为其长和宽相等的矩形。尽管恒等关系可以通过普通的约束关系处理，但是它应用的广泛性和特殊的语义性质，有必要进行特殊处理。这样不仅可以避免通用约束问题求解器带来的低效性，更重要地是它可以直接用来降低问题的计算规模。因为两个相等的变量必须取相同的值，所以可以在计算上将其视为同一个变量。这就是单元共享策略的基本思想。

实现单元共享策略的一种简单方法是将两个相等的变量用一个新的变量代替，但这导致约束网络的全局变量代换。同时，考虑到有些相等关系是在进行了某些赋值假设后作出的，因而必须保存原有的未等同前的变量信息。因此我们采用了二叉树表示。对等式 $x=y$ ，如果 x, y 都已赋值，则判断其是否相等，并返回其真值。如果 x 已赋值而 y 没有赋值，则看 x 的值是否落在 y 的域中。如果不是，则返回不一致（失败）信息。否则，将 x 的值赋于 y 。如果 x, y 都未赋值，则生成一个新的变量节点 $\text{com}(x, y)$ 称为 x 与 y 的公共节点，它的两个分支节点分别为 x 和 y 。 $\text{com}(x, y)$ 的域为 x 和 y 的交集。如果该交集为空，则返回不一致（失败）信息。 x 和 y 与别的节点的约束联结，都被合并到 $\text{com}(x, y)$ 中。以后对 x 或 y 的所有操作，

都落实到 $\text{com}(x, y)$ 上。

一般情况下，当一个变量涉及更多的等式约束时，例如等式集

$$\{x=y, y=z, x=u\}$$

将形成较大的二叉树。

所以一个二叉树的根节点汇集了其所有分支节点的所有信息。它成为所有该二叉树叶节点的代理单元。而当前所访问的变量节点集合为所有这样二叉树的根节点的集合（包括孤立节点）。

对于树搜索而言，必须考虑等式是在某个假设赋值之后作出的情形。当该假设赋值被撤消以后，其后所断言的等价关系也不再成立，应当撤消。其办法是撤消其对应的公共节点。该等式推理器提供对任意两个同类型变量之间相等关系的询问，即使它们未赋值。这种策略具有如下优点，当然这些优点可能是相互关联的。

(1) 由于约束搜索的复杂性主要取决于约束变量的个数，而单元共享策略将两个或更多具有恒等关系的变量合并成一个变量。因而降低了搜索空间。

(2) 使得约束推理器在变量未赋值的情形下也能发现与恒等关系有关的不一致性。例如对约束

$$\{x=y, y=z, x \neq z\}$$

即使不对 x, y, z 赋值系统也能检查出该约束集的不一致性。

(3) 使得约束传播可以提前进行。恒等关系的单元共享策略减少了未知变量的个数，也减少了一个约束关系中未知变量的个数。一般而言，只有当一个约束关系中未知变量的个数足够少时，约束传播才能够得以进行。约束传播的提前进行相当于在搜索树更靠近根节点的部分进行剪枝，因而大大减小了搜索空间。

例如对一个包含三个枚举变量 x, y, z 的约束关系 $r(x, y, z)$ ，当 x 例示以后，如果 y 与 z 为不同的变量，一般而言可能对 y 或 z 的域进行一致性求精。假如存在约束关系 $y=z$ ，则相当于实际未赋值的变量只有一个。因而约束传播得以进行。

(4) 使得约束语言具有模式匹配等符号处理的能力。

为了说明这种策略的优点，考虑由 COPE 语言实现定性物理中的定性加运算的例子。一个在实数域上变化的物理量可以用定性变量来刻画其正负性。一个定性变量取正(pos)、负(neg)、零(zero) 三个值：

```
enum qualitative {pos, neg, zero};
```

定性值的加运算可如下定义：

```
qualsum (x, y, z)
enum qualitative x, y, z;
{
    if (x=y) z=x;
    if (x=zero) z=y;
    if (y=zero) z=x;
}
```

如果我们有约束程序

```
main
{
    enum qualitative u, v, w;
    qualsum(u, v, w);
    qualsum(u, w, v);
}
```

$$\begin{aligned} & w \neq \text{zero}; \\ & \} \end{aligned}$$

则不进行任何试探性搜索，就能求得唯一解，该解为 $\{u=\text{zero}, v=\text{zero}, w=\text{zero}\}$ 。因为由 $\text{qualsum}(u, w, v)$ 及 $w \neq \text{zero}$ ，得 $u=v$ ，该等式使得 $\text{qualsum}(u, v, w)$ 推出 $u=w=\text{zero}$ ；从而 $v=u=\text{zero}$ 。

3.10.2 区间传播

除了等式关系，数域上最通常的关系是不等式，特别是机械电子设备的分析与设计中，不等式的应用尤为重要。不等式表示最基本的推理形式是用于对变量值的测试。即已知变量的值，计算并检查变量的值是否满足该不等式。单纯这种计算方式只能用于约束满足的生成测试策略。而这种策略效率是最低的。这里我们实现了较强形式的不等式推理。

比较常用的不等式的一种推理形式是区间推理，即约束网络上的区间传播。给定一些变量的区间限制，由变量间的序关系，推出对另一些变量的区间限制。例如，设有约束 $x > y$ ，且变量 x 与 y 的区间分别为 $[l_x, g_x]$ 与 $[l_y, g_y]$ ，则对该约束进行约束传播后 x 与 y 的新的区间为

$$\begin{aligned} & [\max(l_x, l_y), g_x] \\ & [l_y, \min(g_x, g_y)] \end{aligned}$$

如果 $g_x < l_y$ ，则意味着矛盾。

这种推理同样可以推广到更复杂的方程式或不等式。考虑方程 $x+y=z$ ，且 x, y, z 的当前区间分别为 $[l_x, g_x], [l_y, g_y], [l_z, g_z]$ 。如果 $l_x + l_y > g_z$ ，或 $g_x + g_y < l_z$ ，则意味着矛盾。否则 z 的新区间值 $[l'_z, g'_z]$ 为

$$\begin{aligned} l'_z &= \max(l_x + l_y, l_z) \\ g'_z &= \min(g_x + g_y, g_z) \end{aligned}$$

x 的新区间值 $[l'_x, g'_x]$ 为

$$\begin{aligned} l'_x &= \max(l_z - g_y, l_x) \\ g'_x &= \min(g_z - l_y, g_x) \end{aligned}$$

y 的新区间值 $[l'_y, g'_y]$ 为

$$\begin{aligned} l'_y &= \max(l_z - g_x, l_y) \\ g'_y &= \min(g_z - l_x, g_y) \end{aligned}$$

3.10.3 不等式图

在很多人工智能应用，如定性推理、时态推理、活动规划中，所关心的是变量间大小的相对关系。因而这种序关系的推理是非常重要的。这实际上是对不等式的公理性质，如恒等关系的公理、偏序关系的公理等进行推理。

将所有形如 $x \leq y$ 与 $x \neq y$ 的不等式组成一个不等式图。关系 $x \geq y$ 表示为 $y \leq x$ ， $x < y$ 表示为 $x \leq y$ 与 $x \neq y$ ， $x > y$ 表示为 $y \leq x$ 与 $x \neq y$ （恒等关系已用单元共享策略实现）。

定义 3.1 递增圈。一个不等式图是一个标记图 $\langle V, E \rangle$, 其中, V 是变量节点的集合。边集 E 是关系表达式 $x \text{ r } y$ 的集合, $x, y \in V, r$ 为关系 $=$ 或 \leq 。网络中的一条递增路径是变量节点的序列 v_1, v_2, \dots, v_l , 使得对任意 $i(i=2, \dots, l), (v_{i-1} \leq v_i)$ 是网络中的一条边。如果 $v_1=v_l$, 则称该路径为一递增圈。

显然, 不等式图具有下列性质:

性质 3.1 一个递增圈如果其中任意两个变量节点(可能相同)之间都不含 \neq 边, 则递增圈中所有的节点都表示相同的变量。

性质 3.2 如果不等式图中存在一个递增圈, 其中某两个变量节点(可能相同)之间含 \neq 边, 则不等式图蕴涵不一致性。

由性质 3.1, 可将一个不等式图进行等价转换, 使其不含递增圈。其办法是对任意不等式图中的递增圈, 都将递增圈中所有的节点按单元共享策略合并为同一个变量节点。将所得到的结果图称为该不等式图的精简图。

由性质 3.2, 我们定义一个不等式图的图不一致性。

定义 3.2 一个不等式图是图不一致的, 当且仅当图中存在一个递增圈, 其中某两个变量节点(可能相同)之间含 \neq 边。

定理 3.1 一个不等式图是图不一致的, 当且仅当其精简图中, 存在一个变量节点, 有一个引向自身的 \neq 边。

基于这两个事实, 不等式推理器实施图遍历过程, 如果发现一个递增圈, 则按单元共享策略将其合并为同一个变量节点。如果一个变量其域为空, 或有一个引向自身的 \neq 边, 则报告不一致性。不难证明上述操作关于偏序关系及恒等关系的完备性, 即给定一个变量间恒等关系、不等关系及偏序关系的集合 \mathcal{W} , 以及有关恒等关系和偏序关系的公理集合 G , 那么, $\mathcal{W} \cup G$ 是不一致的, 当且仅当 \mathcal{W} 所对应的不等式图是图不一致的。

与演绎方法相比较, 这种图操作方法有效得多。因为图遍历过程仅具有线性复杂性。

3.10.4 不等式推理

不等式图体现了不等式的结构性性质。当不等式图中的变量节点被赋予区间限制时, 则除了不等式图本身的操作之外, 还在不等式图上进行约束传播。所有这些操作, 都是在这种不等式网络接收到一个外部输入后引发的。这些外部输入包括: 向网络加入一个关系表达式 $x \text{ r } y$ 。其中 x 与 y 为变量或常量, 但至少有一个为变量。 r 为 $=, \neq, <, \leq, >, \geq$ 之一。

如果 r 为 $=$, 若 x 与 y 都为未赋值的变量, 则按单元共享策略将其合并为同一个变量节点, 并检查是否生成递增圈, 进行可能的不等式图精简与一致性检查。若新的节点的区间比 x 或 y 的区间有所减小, 则沿着不等式网络进行区间传播。如果二者都为已赋值的变量或常量, 则判定其值是否相等; 如果二者有且仅有一个未赋值的变量, 若另一个自变量的值落在该变

量的域中，则将值赋予该变量，并进行约束传播；否则，报告不一致性。

如果 r 为 \leq ，若 x 与 y 都为已赋值的变量，则判定 \leq 关系是否成立；否则，对 x 与 y 的区间进行一致性限定。若限定的结果 x 或 y 的区间有所减小，则沿着不等式网络进行区间传播。如果二者都为未赋值的变量，则向不等式网络加入一条 \leq 边，并检查是否生成递增圈，进行可能的不等式图精简与一致性检查。

如果 r 为 \neq ，则当 x 与 y 都为已赋值的变量时，判定 \neq 关系是否成立。

对 $x < y$, $x > y$ 与 $x \geq y$ 则将其分别等价转化为 $(x \leq y) \wedge (x \neq y)$, $(y \leq x) \wedge (x \neq y)$, $y \leq x$ 。

不等式推理器提供对任意两个同类型变量之间不等关系与大小关系的询问，即使它们未赋值。这种将符号推理与区间传播结合的方法，不仅仅是将不等关系作为内部谓词而减小推理代价，更重要的是它消除了冗余性，降低了问题求解的规模，而且作为一个一般的不等式推理器，能够在变量未赋值甚至未加区间限制的情况下对不等式进行符号推理。例如，系统能够检查出不等式集合

$$\{x \leq y, y \leq z, x > z\}$$

蕴涵的不一致性。这使得系统具有传播序关系的功能。这种功能在定性推理中得到广泛应用。通过实现等式与不等式推理的符号操作，从而增加了推理的深度，避免了更多的生成测试，降低了搜索空间。

3.11 约束推理系统 COPS

约束推理系统的一个主要功能是为用户提供通用而有效的约束推理机制。该种推理机制要克服由于不确定性所引起的搜索问题。约束传播也正是降低不确定性的一种技术。从语言表示上，我们认为应当将不确定的成分显式地标识出来，而将其局限于数据部分。这样数据的相关性成为可识别的，数据域是可操作的，从而可以使用约束传播与智能回溯来减少不确定性，缩小搜索空间。同时这种常规结构也改善了代码的可读性和语言的易学性，以便使用约束传播、智能回溯等专门的搜索技术去处理。逻辑程序设计语言 Prolog 用 Horn 子句上的归结证明法这种试探性求解技术来作为统一的计算机制，虽然形式简单优美，但实用很困难。

在约束推理系统 COPS 中，约束程序设计语言 COPS 将面向对象的技术、逻辑程序设计、产生式系统与约束表示结合起来，在形式上吸收了面向对象的程序设计语言的基本形式，内部求解时采用约束传播和启发式机制，使说明性约束表达式与类型层次相结合，实现结构化的知识封装[史忠植 1996]。COPS 语言具有以下特点：

- (1) 将类型层次与约束表示结合起来；
- (2) 实现默认约束推理；
- (3) 实现条件约束及常规程序设计的其它成分；
- (4) 实现有效的约束推理。

1. 约束与规则

约束是谓词表达式：

$$P(t_1, \dots, t_n)$$

其中 t_1, \dots, t_n 是项, 典型情况它包含变量; P 是谓词符号, 谓词可以是内部函数, 如 `sum`, `times`, `eq(equal)`, `neq(not equal)`, `ge(great than or equal to)`, `gt(great than)`, 也可以由用户定义。

条件约束具有下面的形式:

```
if {
    condition1: constraint1;
    ...
    conditionn: constraintn
}
```

其中, `condition1, ..., conditionn` 是布尔表达式。 `constraint1, ..., constraintn` 是一个约束, 或者是一个具有大括号 `{ }` 的约束表。

规则用来定义新的函数、方法、谓词, 或者可以将约束加到对象上。规则的形式是

RULE [class::] 谓词(变量或常量表) (布尔表达式)

```
{
    约束1;
    ...
    约束n;
    CASE
    布尔表达式1: 约束1;
    ...
    布尔表达式m: 约束m;
}
```

例如:

```
RULE multiple(INTEGER: *x, INTEGER: y, INTEGER: z) (neq(y, 0))
{
    equal(x, divide(z, y));
}
```

定义了三个变量 x, y, z 之间的约束关系: $x = z \div y$ 。

2. 类定义

在 COPS 系统中, 问题领域中的实体被定义为类, 实体的内部属性及它们之间的关系都被封装在类中, 多个具有一定关系的实体又可封装在一个更高层次的类中。COPS 的类的定义与 C++ 的类定义很相似:

```
CLASS [类名][:超类名]
{
    //属性定义
    数据类型: 属性名;
    ...
}
```

```

//规则定义
规则名;

...
//函数定义
函数名;

...
//方法定义
方法名;

...
}

```

整个 COPS 程序就是由类的定义和规则组成。COPS 语言保持了关系式说明型语言的风格, 同时提供类、方法等面向对象成分。这样, 既增强了程序设计的规范性与灵活性, 又提高了程序的易用性和可读性。COPS 语言可以使用户集中于问题本身的描述, 不必关心问题求解的细节。由于 COPS 语言成分与常规的面向对象语言 (C++) 非常类似, 整个描述 直观、清晰, 很容易使用, 而且可以充分利用类的封装性和继承机制进行扩充和复用。COPS 系统已经成功地进行 了电路的模拟。

3. COPS 的约束推理

在 COPS 系统中, 约束推理主要依靠产生式的组合和约束传播, 也具有排序条件重写系统 (ranked conditional term-rewriting system), 利用问题求解状态信息、默认规则和假设推理、分区传播等启发式特点。下面给出 COPS 系统的核心算法。

算法 3.9 COPS 系统的核心算法 main-COPS。

```

procedure main_COPS
{
    1. 调用 yacc 分析程序, 生成内部结构;
    2. 初始化;
        建立 COPS 常数 trueNode;
        全局变量分配存储空间;
    3. 解释具有内部结构的程序;
    4. 对尚未求解的约束和变量建立约束网络;
    5. while 触发约束网络中的约束
        解释触发约束.
}

```

上述算法中的 yacc (Yet Another Compiler Compiler) 是又一个编译程序的编译程序, 它 把一上下文无关文法转换为一种简单自动机的一组表格, 该自动机执行一个 LR 语法分析程 序。输出文件为 y.tab.c, 必须由 C 编译程序编译, 产生程序 yyparse。该程序必须与词法分 析程序 yylex, 以及 main 和出错处理程序 yyerror 一起安装。

算法中的解释器如下:

```

Interpreter:
{
    switch (constraint type)
    case Constant:
        return Constant;
    case global variable:

```

```

        interpret global variable:
case local variable or argument:
    interpret local variable or argument:
case object-attribute pair:
    interpret object-attribute pair:
case function call:
    interpret function call:
case method call:
    interpret method call:
case CASE expression:
    interpret CASE expression:
    ...
default:
    report error
}

```

COPS 系统充分利用类的封装性和继承机制进行扩充和复用，通过类的成员函数的重载，高效灵活实现约束求解。我们还可以通过设计新的求解类，在 COPS 系统中加入多种约束求解方法，改进原有系统求解策略单一的弱点。目前，要解决的主要问题是在不同类之间的进行消息传递、对共享变量进行值传播和一致性维护的过程中如何避免组合爆炸。今后的工作打算在现有系统的基础上将约束技术运用于多主体系统中，解决多主体系统中的协作和协商问题；将约束推理应用于智能决策支持系统中的多目标问题求解。

3.12 ILOG Solver

约束程序是关于约束的计算系统，它的输入是一组约束条件和需要求解的若干问题，输出问题的解决方案。至于具体解决问题的算法等都是约束程序设计语言的基本功能。程序员所要面对的，就是如何把问题描述为一组约束构成的模型，而描述的语言可以很接近自然语言。如果把问题的解决方案也看作是一种约束，那么问题的求解就是求得一个或若干个约束，他们每一个都是这一组给定约束的充分条件，也就是说，求得的这些约束，满足这组给定的约束。于是，约束程序设计便可以称为面向约束的程序设计方法。

ILOG 公司是法国优化、互动图像界面以及商业规则应用领域的软件组件供应商，成立于 1987 年。在过去的时间内，ILOG 公司不断进行企业软件组件和服务的开发与创新，使得客户优化了业务处理的灵活性，并且提高了这些公司的运营效率。超过 1000 家的全球公司和 300 多家软件供应商使用 ILOG 的产品。随着中国经济的飞速发展，ILOG 公司看到了中国市场的广阔发展前景，于 2002 年 8 月设立了北京代表处，统领包括香港、澳门、台湾地区在内的大中华区市场。

约束规划是基于约束规则的计算机系统的程序，约束规划的概念是描述问题的约束来解决问题。结果是找到让所有的约束满意的方案。规划调度的实施的关键是基于约束规则，基于约束自动的调配资源，优化计划，来达到你所需要的计划目标。对离散的制造行业解决复杂的加工过程如多工序，多资源等；对重复式或流程式的制造行业解决顺序问题如优化排序等（Flowshop 调度）。ILOG Solver 是嵌入过程性语言的约束程序设计语言，他将面向对象程序设计和约束逻辑程序设计结合起来，包含逻辑变量，通过增量式约束满足和回溯实现问题求解。ILOG Solver 中主要语言成分如下：

```
variables : C++ object /* 变量 */
```

```

integer variable      CtIntVar
floating variable    CtFloatVar
boolean variable     CtBoolVar

Memory Management /* 存储管理 */
new:
delete:

Constraints /* 约束 */
    CtTell(x == (y + z));

    Basic constraints: =, ≤, ≥, <, >, +, -, *, /, subset, superset, union, intersection,
    member, boolean or, boolean and, boolean not, boolean xor;
    CtTell((x==0) || (y==0));
    CtlfThen (x < 100, x = x+1);

Search /* 搜索 */
    CTGOALn: how to execute          CTGOAL1(CtInstantiate, CtIntVar* x){
        CtInt a = x->chooseValue();
        CtOr(Constraint(x == a),
            CtAnd(Constraint(x != a),
                CtInstantiate(x)));
    }

Schedule /* 调度 */
    CtSchedule class
        Global object: time original ---timeMin
                    time horizon ---timeMax

Resources /* 资源 */CtResource
    CtDiscreteResource
    CtUnaryResource
    CtDiscreteEnergy
    CtStateResource

Activities /* 工序 */
    CtActivity class
        CtIntervalActivity

```

一个动作被定义为开始时间、结束时间、时间跨度、工序要求、提供、消费和生产资源。

约束规划的开发已经吸引各个领域的专家的高度注意，因为它有潜力解决现实中非常难的问题。无论我们是用先进的遗传算法，还是用人机交互式的仿真方法，都需要对制造业的复杂约束，多目标优化，大规模的搜索和车间生产的不确定性的问题进一步研究，以适用实际需要。这里采用基于事件的调度方法，即至少一个资源是空闲的，二个或多个工序能用于这个资源，工序选择规则 OSR (Operation Selection Rule) 决定那一个工序被加载。这就是决定计划结果质量好坏的关键因素。独立的工序选择规则详细介绍如下：

- (1) 最早完成日期：选择最早完成的工序（也许是订单完成日期）
- (2) 最高优先级第一：选择最高优先级（最低值）的工序
- (3) 最低优先级第一：选择最低优先级（最高值）的工序
- (4) 最高订单属性字段：选择最高（最大）订单属性字段的工序
- (5) 最低订单属性字段：选择最低（最小）订单属性字段的工序

- (6) 动态最高订单属性字段：选择动态最高（最大）订单属性字段的工序
- (7) 动态最低订单属性字段：选择动态最低（最小）订单属性字段的工序
- (8) 计划档案订单：选择订单里出现先到先服务的工序
- (9) 关键率：选择最小关键率的工序。

关键率=剩余计划工作时间/（完成日期-当前时间）

- (10) 实际关键率：选择最小实际关键率的工序
实际关键率=剩余实际工作时间/（完成日期-当前时间）
- (11) 最少剩余工序（静态）：选择最少剩余工序时间的工序
- (12) 最长等待时间：选择最长等待时间的工序
- (13) 最短等待时间：选择最短等待时间的工序
- (14) 最大过程时间：选择最大过程时间的工序
- (15) 最小过程时间：选择最小过程时间的工序
- (16) 最小工序闲散时间：选择最小工序闲散时间的工序。

订单任务的闲散时间=任务剩余完成时间-剩余工作时间

工序闲散时间=任务闲散时间/完成任务的剩余工序数

- (17) 最小订单闲散时间：选择最小订单任务的闲散时间的工序
- (18) 最小工作剩余：选择所有需要完成订单的最小剩余过程时间的工序。

资源选择规则（Resource Selection Rule）选择工序加载到资源组内的哪一资源。

- (1) 最早结束时间：选择将要最先完成工序的资源
- (2) 最早开始时间：选择将要最先开始工序的资源
- (3) 最迟结束时间：选择将要最迟完成工序的资源
- (4) 与前工序一样：选择被用于前一工序的资源
- (5) 非瓶颈最早开始时间：选择将要最早开始工序的非瓶颈资源

相关选择规则是如果选择一工序选择规则，就自动的选择相应的资源选择规则。

(1) 系列顺序循环：选择同样或下一个最高（最低）系列值的工序。当没有最高值的工序，顺序将相反，选择最低的工序。

- (2) 系列降顺序：选择同样或下一个最低系列值的工序
- (3) 系列升顺序：选择同样或下一个最高系列值的工序
- (4) 最小准备系列：选择最小准备时间及最近的系列值的工序。
- (5) 最小准备时间：选择最小准备或换装时间的工序

(6) 定时区的系列顺序循环：选择同样或下一个最高（最低）系列值工序。且只考虑在特定的时区里的订单完成日期里的工序。当没有最高值的工序，顺序将相反，选择最低的工序。

(7) 定时区的系列降顺序：选择同样或下一个最低系列值工序。且只考虑在特定的时区里的订单完成日期里的工序。

(8) 定时区的系列升顺序：选择同样或下一个最高系列值工序。且只考虑在特定的时区里的订单完成日期里的工序。

(9) 定时区的最小准备系列：选择最小准备时间及最近的系列值的工序。且只考虑在特定的时区里的订单完成日期里的工序。

(10) 定时区的最小准备时间：选择最小准备或换装时间的工序，且只考虑在特定的时区里的订单完成日期里的工序。

下面以房屋装修为例（图 3.1），采用 ILOG Scheduler 给出规划方案，说明约束问题求解的原理。假定任务开始每天的开销是 1000 元，资金总额为 20000。到工程进行到第 15 天，可以增加 9000 元。

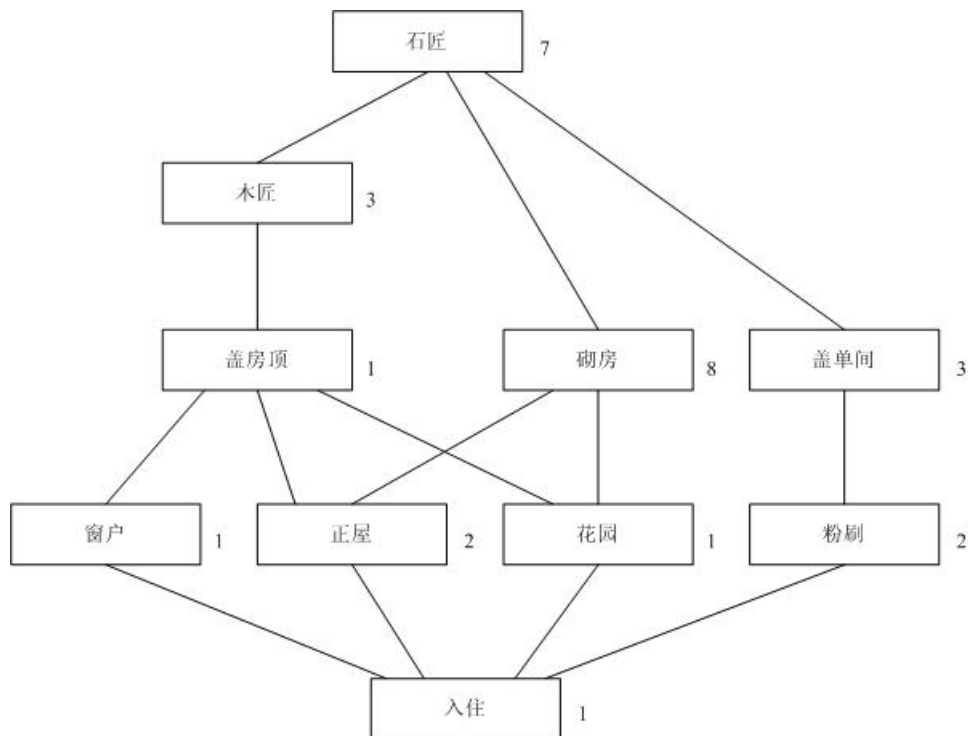


图 3.1 房屋装修工序

采用 ILOG Scheduler 进行规划, 约束程序如下:

```

CtSchedule * schedule =
    new CtSchedule(0, horizon);
// 创建具有给定期限的工序.
CtIntervalActivity* act =
    new CtIntervalActivity(schedule, duration);
//规定工序 act1 和 act2 之间的顺序约束.
act2->startsAfterEnd(act1,0);

//创建有限资金 capacity 29000 的总预算.
CtDiscreteResource* res =
    new CtDiscreteResource(schedule,
        CtRequiredResource,
        capacity);
// 说明开始 15 天里,能用的资金 cap 为 20000.
res->setCapacityMax(0,date,cap);
// 说明一个工序 act 消耗的资源 res 为 c 单位.
act->consumes(res, c);

CtBoolean IsUnScheduled(CtActivity* act){
// 如果工序 act 没有固定开始时间则返回 True.
if (act->getStartVariable()->isBound())
    return CtFalse;
else

```

```

        return CtTrue;
    }
    CtBoolean IsMoreUrgent(CtActivity* act1,
                          CtActivity* act2){
    // 如果工序 act1 比 act2 紧迫, 则返回 true .
    // 如果工序没有限制, 则返回 true .
    if (act2 == 0)
        return CtTrue;
    else if (act1->getStartMax() < act2->getStartMax())
        return CtTrue;
    else
        return CtFalse;
    }

    CtActivity* SelectActivity(CtSchedule* schedule){
    // Returns the unscheduled activity with the smallest latest
    // start time. Returns 0 if all activities are scheduled.
    CtActivity* bestActivity = 0;
    //Creates an iterator to iterate on all activities.
    CtActivityIterator* iterator(schedule);
    CtActivity* newActivity;
    while(iterator.next(newActivity))
        if((IsUnScheduled(newActivity))
            && (IsMoreUrgent(newActivity, bestActivity)))
            bestActivity = newActivity;
    return bestActivity;

    void SolveProblem(CtSchedule* schedule){
    // Solve the problem assuming constraints have been posted.
    CtActivity* act = SelectActivity(schedule);
    while (act !=0) {
        act->setStartTime(act->getStartMin());
        act = SelectActivity(schedule);
    }
    }
}

```

ILOG Scheduler 求解, 得到图 3.2 所示的装修工序规划图。

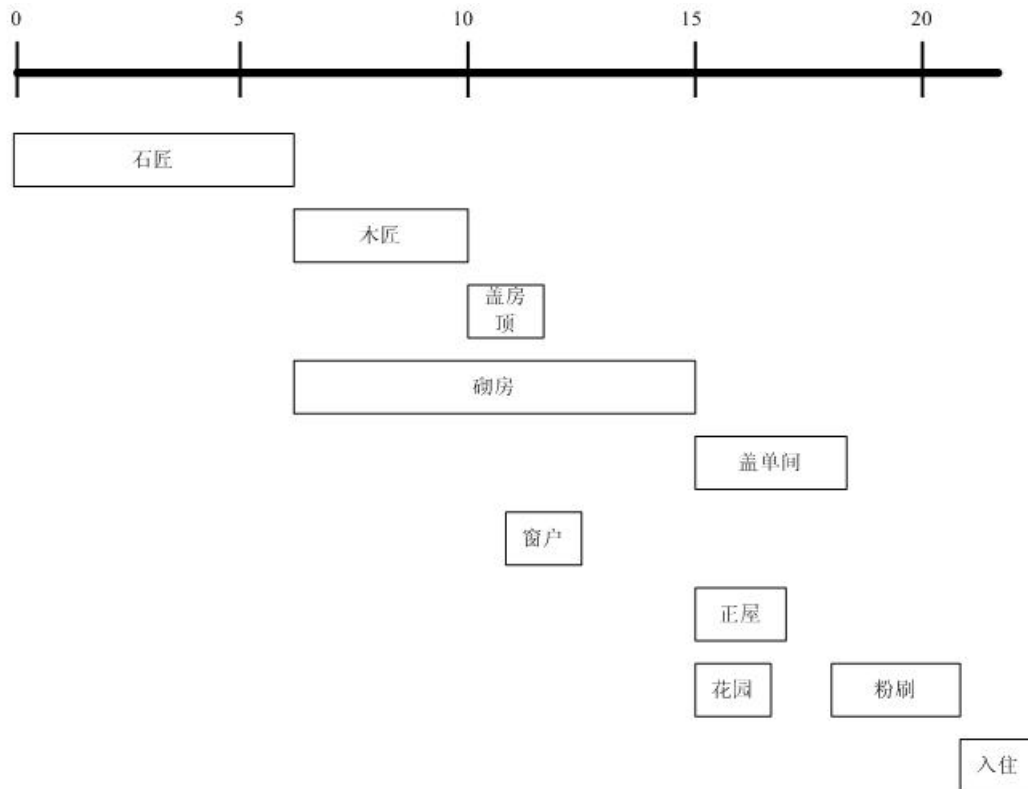


图 3.2 房屋装修工序规划图

习 题

12. 什么是约束满足问题？约束推理可以分为哪几种？
13. 什么是弧一致性？并举例说明它的非对称性。
14. 请给出约束传播 AC-1 和 AC-3 算法，并比较它们的异同之处。
15. 试用程序设计语言编写基于影响的回跳算法 IBMD，并用 N 皇后问题测试，与其他约束算法进行比较。
16. 如何使约束推理系统 COPS 既能求解符号推理问题，又能求解数值问题？
17. 用 ILOG Solver 语言写一个车间调度系统。

第四章 定性推理

定性推理(qualitative reasoning)是从物理系统、生命系统的结构描述出发,导出行为描述,以便预测系统的行为并给出原因解释。定性推理采用系统部件间的局部结构规则来解释系统行为,即部件状态的变化行为只与直接相邻的部件有关。

4.1 概 述

人工智能的定性推理理论起源于对物理现象的研究,早期的工作常常是针对一物理过程如动力学、流体力学、热流等问题来讨论的。1952年,Simmons 提出定性分析的因果关系。1977年,Rieger 发表了因果仿真的论文[Rieger 1977]。1984年,“Artificial Intelligence”杂志第24卷出版了定性推理专辑,刊载了 de Kleer, Forbus 和 Kuipers 对定性推理奠基性的文章,这标志着定性推理开始走向成熟。1986年,Iwasaki 和 Simmons 发表了“Causality in Device Behavior”的文章[Iwasaki & Simmons 1986]。十年来,这些基本方法对定性推理的研究和应用起着重要的作用,使定性推理的研究成为人工智能中富有成果的领域之一。1993年,“Artificial Intelligence”杂志第59卷又发表了一组文章,回顾十年前这几位定性推理奠基人所做的工作。

对物理系统不同的结构描述,便提出了不同的定性推理方法。常用的有 de Kleer 的定性模型方法[de Kleer 1984]、Forbus 的定性进程方法[Forbus 1984]、Kuipers 定性仿真法[Kuipers 1984]。de Kleer 的定性模型方法所涉及的物理系统是由管子、阀门、容器等装置组成,约束条件(定性方程)反映在这些装置的连接处,依定性方程给出定性解释。在 Forbus 的定性进程方法中,一个物理系统的变化是由进程引起的,一个物理过程由一些进程来描述。Kuipers 直接用部件的参量作为状态变量来描述物理结构,定性约束直接由物理规律得到,把一个参量随时间的变化视作定性的状态序列,求解算法是从初始状态出发,生成各种可能的后续状态,进而通过一致性过滤,重复这过程直到没有新状态出现。

除了上面这三种基本方法外,还有其它的研究工作。如 Davis 提出从结构描述出发进行故障论断的方法。Reiler 提出从基本原理出发进行故障诊断的方法。Williams 把定量运算和定性推理相结合建立了一个混合代数系统 Q1,并讨论了它的代数性质,他还用定性方程实现了设计方面的定性推理。Iwasaki 和 Simmons 把经济学、热力学中所用的因果关系形式化特征和相对静止的方法用于定性因果分析,对因果关系给出了形式化定义。Weld 在分子生物学中设计了定性模拟程序,用聚类方法找出了重复出现的循环,通过对循环的分析确定系统的最后状态,Weld 还讨论了非连续量的情况。

4.2 定性推理的基本方法

人类对物理世界的描述、解释,常是以某种直观的定性方法进行的,很少使用微分方程及具体的数值描述,如人们在骑自行车时,为了避免摔倒和撞车,并不需要使用书本上的运动方程,而是针对几个主要参量的变化趋势给予粗略的、直观的,但大体上准确的描述,这就够了。

一般分析运动系统行为的标准过程可分为三个步骤：

- (1) 决定描述对象系统特征的量。
- (2) 用方程式表示量之间的相互关系。
- (3) 分析方程式，得到数值解。

这类运动系统行为的问题用计算机进行求解时，将面临如下三个问题：

- (1) 步骤(1)(2)需要相当多的知识，并且要有相应的算法。
- (2) 有的场合对象系统的性质很难用数学式子表示。
- (3) 步骤(3)得到了数值解，但是对象系统的行为并不直观明了。

为了解决第二、第三个问题，定性推理一般采用下列分析步骤：

- (1) 结构认识：将对象系统分解成部件的组合。
- (2) 因果分析：当输入值变化时，分析对象系统中怎样传播。
- (3) 行为推理：输入值随着时间变化，分析对象系统的内部状态怎样变化。
- (4) 功能说明：行为推理的结果表明对象系统的行为，由此可以说明对象系统的功能。

定性推理的观点大体上可这样来理解：

- 忽略被描述对象的次要因素，掌握主要因素简化问题的描述。
- 将随时间 t 连续变化的参量 $x(t)$ 的值域离散化为定性值集合，通常变量 x 的定性值 $[x]$ 定义为

$$[x] = \begin{cases} - & \text{当 } x < 0 \\ 0 & \text{当 } x = 0 \\ + & \text{当 } x > 0 \end{cases}$$

- 依物理规律将微分方程转换成定性(代数)方程，或直接依物理规律建立定性模拟或给出定性进程描述。
- 最后给出定性解释

4.3 定性模型推理

de Kleer 注意到符号计算和推理是理解人们周围的物理世界最理想的工具。1974 年，de Kleer 参加了为期半年的“混淆研讨会(confusion seminar)”。会上讨论了一系列人们熟悉而又令人困惑的问题，如反弹球、滑轮、钟摆、弹簧等问题。研讨会的目的是研究人类思考推理的过程，而 de Kleer 却带着完全不同的课题离开研讨会。他发现对于大多数例子，传统的数学、物理公式都没有用或不必要，许多令人迷惑的问题只需要简单的定性推理，至多只用一两个极简单的方程就可得到令人满意的解。

de Kleer 研究解决经典物理问题需要哪些知识及如何建立问题求解系统。他提出的定性模型方法所涉及的物理系统是由管子、阀门、容器等装置组成，约束条件(定性方程)反映在这些装置的连接处，依定性方程给出定性解释。

为将代数方程、微分方程定性化，首先需定义变量的定性值集合以及相应的定性运算。

定性值集合是一个离散集合，其元素是由对数轴的划分而得到的，通常把数轴 $(-\infty, \infty)$ 划分成 $(-\infty, 0)$ ， 0 ， $(0, \infty)$ 三段，规定定性值集合为 $\{-, 0, +\}$ ，变量 x 的定性值 $[x]$ 如下定义：

$$[x] = \begin{cases} - & \text{当 } x < 0 \\ 0 & \text{当 } x = 0 \\ + & \text{当 } x > 0 \end{cases}$$

另外用 ∂x 表示 dx/dt 的定性值，也即

$$\partial x = \left[\frac{dx}{dt} \right]$$

定性值的加、乘运算分别以 \oplus 、 \otimes 表示，可按下述定义：

$y \backslash x$	-	0	+
-	-	-	?
0	-	0	+
+	?	+	+

$$[x] \oplus [y]$$

$y \backslash x$	-	0	+
-	+	0	-
0	0	0	0
+	-	0	+

$$[x] \otimes [y]$$

其中：符号？表示不确定或无定义。下面给出 \oplus 和 \otimes 的运算规则。设 e_1, e_2 是公式，则有：

$$[0] \oplus [e_1] \Rightarrow [e_1]$$

$$[0] \otimes [e_1] \Rightarrow [0]$$

$$[+] \otimes [e_1] \Rightarrow [e_1]$$

$$[-] \otimes [e_1] \Rightarrow -[e_1]$$

使用下列规则，可将运算符+、 \times 转换成 \oplus 、 \otimes ：

$$[e_1 + e_2] \Rightarrow [e_1] \oplus [e_2]$$

$$[e_1 \times e_2] \Rightarrow [e_1] \otimes [e_2]$$

由这些运算规则不难将通常的代数方程、微分方程化成定性方程。由定性方程给出解释便是进行定性推理的过程。这里以压力调节器定性分析为例，说明定性模型推理的方法。

压力调节器是通过弹簧来控制阀门流量，以使流量为某一设定值而不受流入的流量和负载变化的影响。根据物理学有

$$Q = CA\sqrt{2\frac{P}{\rho}} \quad P > 0$$

$$\frac{dQ}{dt} C\sqrt{2\frac{P}{\rho}} \frac{dA}{dt} + \frac{CA}{\rho} \sqrt{\frac{\rho}{2P}} \frac{dP}{dt}$$

其中 Q 是通过阀门的流量, P 是压力, A 是阀门开启的面积, 而 C 是常系数, ρ 是流体的质量密度。按照运算和转换规则而得到定性方程:

$$[Q] = [P]$$

$$\partial Q = \partial A + \partial P \quad (\text{如果 } A > 0)$$

根据一致性、连续性等物理规律还可建立有关的定性方程。由这些定性方程便可得出定性解释, 描述调节器有三个特殊的状态, 即开、关、工作状态:

OPEN 状态	$A = A_{\max}$	定性方程	$[P] = 0$	$\partial P = 0$
WORKING 状态	$0 < A < A_{\max}$	定性方程	$[P] = [Q]$	$\partial P + \partial A = \partial Q$
CLOSED 状态	$A = 0$	定性方程	$[Q] = 0$	$\partial Q = 0$

除了可以讨论每个状态内的定性分析还可讨论各状态间转换的定性分析。de Kleer 建立的 ENVISION 系统是使用约束传播与生成测试方法来求解定性方程。

4.4 定性进程推理

Forbus 提出的定性进程方法把物理现象视作由一些相关的进程来描述, 每个进程由一组个体、前提条件、数量条件、参数关系和影响来描述, 推理过程是从已知的进程表中依次选出一些可用的进程来描述一个物理过程。定性进程理论中有关定性物理的关键思想如下:

(1) 组织原则为物理进程。本体论在知识的组织上起着重要作用。在人们进行物理系统推理时, 物理进程非常直观, 用它组织物理领域的理论是合理的。

(2) 用顺序关系表示数值。重要的性质差别常由比较而来。例如, 当压力和温度不同时产生流动; 当温度到达某一界值时会发生相变等。在很多情况下, 用一套序数关系表示数值更自然。

(3) 单一机制假设。物理进程被看作是产生变化的机制。这样, 任何变化必须解释为某些物理进程的直接或间接的影响。进程本体论为定性物理理论的因果性打下了基础。

(4) 组合的定性数学。人们进行复杂系统推理时, 使用部分信息并进行组合。

(5) 清晰的表示及关于模型化假设的推理。明确地表示某些特定知识的适用条件, 并从领域理论中为特定系统建模成为定性物理的中心任务。

一个物理系统的变化是由进程引起的, 一个物理过程由一些进程来描述, 这就是定性推理进程方法的基本观点。下面介绍在定性进程推理中的量空间和进程的描述。

1. 量空间

(1) 时间由区间表示, 区间之间的关系有前、后、相等。两个区间可以相连, 瞬间认为

是极短的区间，持续时间为 0。

(2) 物体的参数称作量，量由其数量和导数组成。

A_n 表示数量的值， A_s 表示数量的符号。

D_n 表示数量导数值， D_s 表示数量导数的符号。

(MQ_t) 表示时刻 t 量 Q 的值。

HAS-Quantity 是谓词，指某物体具有某参数。

(3) 一个量的所有可能取值构成量空间，量空间的元素间有半序关系。

如果 $Q_1 = f(Q_2)$ 是单调上升的，说 Q_1 与 Q_2 定性成正比例，记作 $Q_1 \propto_{Q^+} Q_2$ 。如果 $Q_1 = f(Q_2)$

是单调下降的，说 Q_1 与 Q_2 定性成反比例，记作 $Q_1 \propto_{Q^-} Q_2$ 。

2. 进程

一个物理进程 P 由一组个体、一组前提条件、一组数量条件、一组参数关系和一组影响组成。一个进程的具体示例称作进程例，用 PI 表示。

所谓影响，是指什么能引起参数的变化。影响有直接影响和间接影响之分。如果在某一时刻有一进程影响量 Q ，则说 Q 受直接影响。如果数 n 直接影响 Q 且影响为正、负、无，则分别记作 $1+(Q, n)$, $1-(Q, n)$, $1\pm(Q, n)$ 。当 Q 是其它量的函数时，称 Q 受间接影响。如定性成比例 \propto_Q 就是间接影响。进程表指一个领域可能出现的全部进程。

在进程推理方法中，一个物理过程可用一些进程来描述。这里以热流进程为例说明该方法的工作原理。

Process heat-flow.	热流进程
Individuals:	一组个体
src an object, Has-Quantity(src, heat)	src 是热源
dst an object, Has-Quantity(dst, heat)	dst 是受热对象
path a heat-path,	path 是热流路径
Heat-connection(path, src, dst)	将 src, dst 连结起来
Preconclitions:	一组前提条件
Heat-Aligned(path)	热流路径安排好
Quantity Conditions:	一组数量条件
A[temperature(src)] > A[temperature(dst)]	src 温度高于 dst 温度
Relations:	一组参量关系
Let flow-rate be a quantity	flow-rate (热流量) 是一个数量
A[flow-rate] > ZERO.	flow-rate 值 > 0
flow-rate \propto_{Q^+} (temperature(src) - temperature(dst))	flow-rate 与 src, dst 的温差定性成比例
Influences:	一组影响
1-(heat(src), A[flow-rate])	flow-rate 的值直接影响 heat(src), 而且是负影响
1+(heat(dst), A[flow-rate])	flow-rate 的值直接影响 heat(dst), 而且是正影响

3. 演绎过程

在进程定性推理中，其演绎过程如下：

(1) 选进程。对一组已知的个体来说，在进程表中依各进程对个体的说明找出可能出现的那些进程例 PI。

(2) 确定激活的 PI。依前提条件、数量条件确定每个 PI 的状态。满足这些条件的为激活的 PI，激活的 PI 叫进程结构。

(3) 确定量的变化。个体的变化由相应量的 D_s 值来表示。量的变化可由进程直接影响，也可由 \propto_q 间接影响。

(4) 确定进程结构变化。量的变化将会引起进程结构的变化，确定这种变化也叫限制分析，这样对一个物理过程的描述便由 (1) 建立的 PI 进入了下一个 PI。

重复(1)－(4)的步骤便可给出一个物理过程的一串进程描述。

限制分析是依 D_s 值来确定量在量空间中的变化。首先在量空间中找出当前值的相邻。如果相邻是限制点，则某些进程便停止，某些进程开始。与限制点有关的半序关系的所有可能变化确定了当前激活进程的变化路径。

这里以锅炉加热过程的进程描述为例，说明进程方法的演绎过程。锅炉有一个装水的容器，加热时盖子密封，热源是火，假设热源的温度不变。当容器内压力超过 p-burst(CAN) 时就会爆炸。

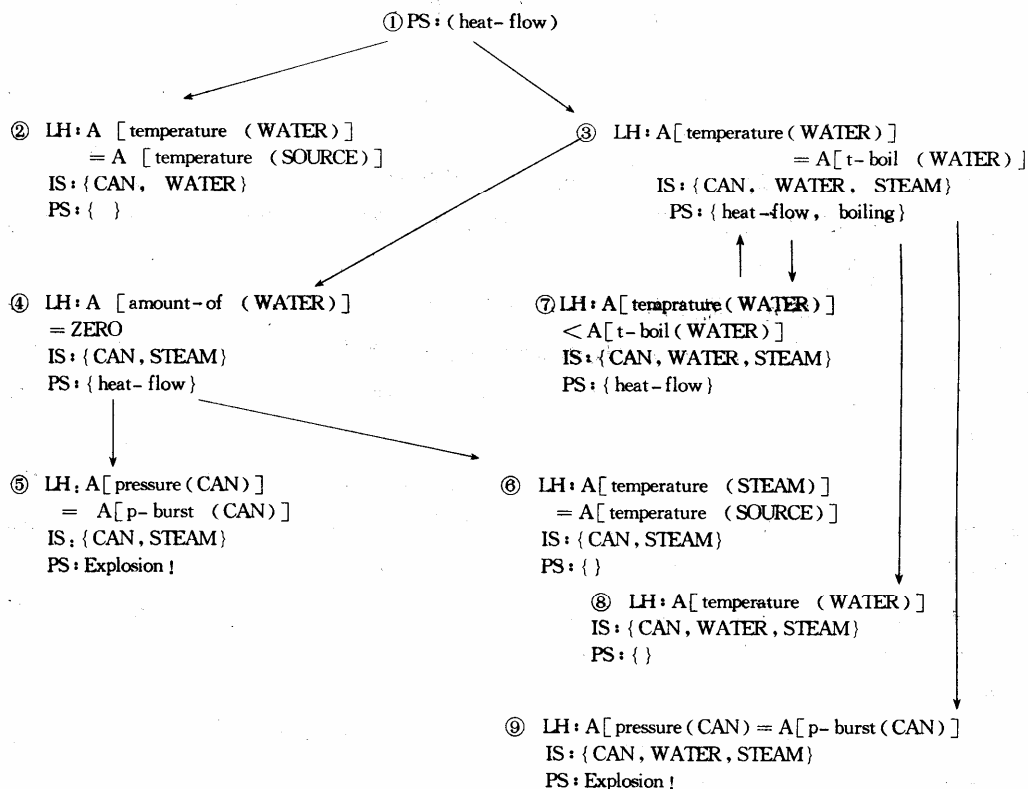


图 4.1 锅炉加热过程的进程描述

图 4.1 给出了一个锅炉加热过程的进程描述，其中 PS 是进程结构，LH 是限制假设(附加的量假设)，IS 是出现的个体。锅炉加热过程的进程解释如下：

从 ①开始，进程结构只有 heat-flow(热流)。由 PS: (heat-flow)可导出 ②和 ③。②

是水温与热源温度相等，这时 heat-flow 进程结束。③ 是水开始沸腾，由 heat-flow 和 boiling 进程描述，这时可发生④，⑦，⑧，⑨。④ 是水烧干了，可导至 ⑤出现爆炸进程，或 ⑥蒸汽温度达到热源温度而结束。⑦ 是水沸腾使压力上升，使水沸点提高，由于水温低于沸点。返回③。⑧ 是水温达到热源温度而结束。⑨ 容器压力过高，用进程 p-burst (CAN) 描述，出现爆炸。

4.5 定性仿真推理

1984 年 Kuipers 发表了“因果性的常识推理：从结构导出行为”论文。这篇论文建立了一种定性仿真推理的框架，简单地给出了从常微分方程的抽象而得的定性结构和定性行为表示方法。随后，1986 年 AI 杂志又刊登了 Kuipers “定性仿真”一文，文中明确了抽象关系，提出用于定性仿真的 QSIM 算法，并用抽象关系证明了其有效性和不完备性。这两篇文章奠定了定性仿真的基础。

定性仿真从结构的定性描述出发来导出行为描述。直接用部件的参量作为状态变量来描述物理结构，定性约束直接由物理规律得到，把一个参量随时间的变化视作定性的状态序列，求解算法是从初始状态出发，生成各种可能的后续状态，进而通过一致性过滤，重复该过程直到没有新状态出现。

定性仿真结构描述由系统的状态参数和约束关系组成。认为参数是时间的可微函数，约束是参数间的二元或多元关系。如速度的导数是加速度。表为 $DERIV(vel, acc)$ 。 $f = ma$ 表为 $MULT(m, a, f)$ ， f 随 g 单调增加表为 $M^+(f, g)$ ， f 随 g 单调减少，表为 $M^-(f, g)$ 。

行为描述关心参量的变化。假设参量 $f(t)$ 是 $[a, b]$ 到 $[-\infty, \infty]$ 的可微函数。 f 的界标值是一个有限集合，至少含有 $f(a), f(b)$ 。集合 $\{t | t \in [a, b] \wedge f(t) \text{ 是界标值}\}$ 的元素称作区别点。

定义 4.1 设 $l_1 < l_2 < \dots < l_k$ 是 $f: [a, b] \rightarrow [-\infty, \infty]$ 的界标值。对任意 $t \in [a, b]$ ， f 在 t 的定性状态 $QS(f, t)$ 规定为有序对 $\langle qval, qdir \rangle$ ，定义如下：

$$qval = \begin{cases} l_j & \text{如果 } f_t = l_j \\ (l_j, l_{j+1}) & \text{如果 } f_t \in (l_j, l_{j+1}) \end{cases}$$

$$qdir = \begin{cases} inc & \text{如果 } f'(t) > 0 \\ std & \text{如果 } f'(t) = 0 \\ dec & \text{如果 } f'(t) < 0 \end{cases}$$

定义 4.2 设 t_i, t_{i+1} 是相邻的区分点，规定 f 在 (t_i, t_{i+1}) 内的定性状态 $QS(f, t_i, t_{i+1})$ 仍为

$$QS(f, t) \quad \text{其中 } t \in (t_i, t_{i+1}) \quad (4.1)$$

定义 4.3 f 在 $[a, b]$ 上的定性行为是 f 的定性状态序列 $QS(f, t_0), QS(f, t_0, t_1), QS(f, t_1), \dots, QS(f, t_n)$ 。其中 $t_i (i=0, 1, \dots, n)$ 为所有的区分点，且 $t_i < t_{i+1}$ ，若 $F = \{f_1, \dots, f_n\}$ ，则 F 的定性行为是

$$QS(F, t_i) = \{ QS(f_1, t_i), \dots, QS(f_n, t_i) \} \quad (4.2)$$

$$QS(F, t_i, t_{i+1}) = \{ QS(f_1, t_i, t_{i+1}), \dots, QS(f_n, t_i, t_{i+1}) \} \quad (4.3)$$

其中 t_i 是 f_1, \dots, f_n 区分点并集的元素。

4.5.1 定性状态转换

在定性仿真中，定性状态转移是经常遇到的。假设 f 是可微函数， f 从一个定性状态转换到另一个定性状态必须遵守介值定理和中值定理。定性状态转换有两类：一类是 P 转换，该类转换是从时间点到时间区间；另一类是 I 转换，它是从时间区间到时间点的转换。下面给出转换表：

P 转换	$QS(f, t_i)$	\Rightarrow	$QS(f, t_i, t_{i+1})$
P1	$\langle l_j, \text{std} \rangle$	\Rightarrow	$\langle l_j, \text{std} \rangle$
P2	$\langle l_j, \text{std} \rangle$	\Rightarrow	$\langle (l_j, l_{j+1}), \text{inc} \rangle$
P3	$\langle l_j, \text{std} \rangle$	\Rightarrow	$\langle (l_{j-1}, l_j), \text{dec} \rangle$
P4	$\langle l_j, \text{inc} \rangle$	\Rightarrow	$\langle (l_j, l_{j+1}), \text{inc} \rangle$
P5	$\langle (l_j, l_{j+1}), \text{inc} \rangle$	\Rightarrow	$\langle (l_j, l_{j+1}), \text{inc} \rangle$
P6	$\langle l_j, \text{dec} \rangle$	\Rightarrow	$\langle (l_{j-1}, l_j), \text{dec} \rangle$
P7	$\langle (l_j, l_{j+1}), \text{dec} \rangle$	\Rightarrow	$\langle (l_j, l_{j-1}), \text{dec} \rangle$
I 转换	$QS(f, t_i, t_{i+1})$	\Rightarrow	$QS(f, t_{i+1})$
I1	$\langle l_j, \text{std} \rangle$	\Rightarrow	$\langle l_j, \text{std} \rangle$
I2	$\langle (l_j, l_{j+1}), \text{inc} \rangle$	\Rightarrow	$\langle l_{j+1}, \text{std} \rangle$
I3	$\langle (l_j, l_{j+1}), \text{inc} \rangle$	\Rightarrow	$\langle l_{j+1}, \text{inc} \rangle$
I4	$\langle (l_j, l_{j+1}), \text{inc} \rangle$	\Rightarrow	$\langle (l_j, l_{j+1}), \text{inc} \rangle$
I5	$\langle (l_j, l_{j+1}), \text{dec} \rangle$	\Rightarrow	$\langle l_j, \text{std} \rangle$
I6	$\langle (l_j, l_{j+1}), \text{dec} \rangle$	\Rightarrow	$\langle l_j, \text{dec} \rangle$
I7	$\langle (l_j, l_{j+1}), \text{dec} \rangle$	\Rightarrow	$\langle (l_j, l_{j+1}), \text{dec} \rangle$
I8	$\langle (l_j, l_{j+1}), \text{inc} \rangle$	\Rightarrow	$\langle l^*, \text{std} \rangle$
I9	$\langle (l_j, l_{j+1}), \text{dec} \rangle$	\Rightarrow	$\langle l^*, \text{std} \rangle$

```

\begin{tabular}{lll}
& QS(f, t_i, t_{i+1}) & \Rightarrow QS(f, t_{i+1}) \\
I1 & \langle l_j, \text{std} \rangle & \Rightarrow \langle l_j, \text{std} \rangle \\
I2 & \langle (l_j, l_{j+1}), \text{inc} \rangle & \Rightarrow \langle l_{j+1}, \text{std} \rangle \\
I3 & \langle (l_j, l_{j+1}), \text{inc} \rangle & \Rightarrow \langle l_{j+1}, \text{inc} \rangle \\
I4 & \langle (l_j, l_{j+1}), \text{inc} \rangle & \Rightarrow \langle (l_j, l_{j+1}), \text{inc} \rangle \\
I5 & \langle (l_j, l_{j+1}), \text{dec} \rangle & \Rightarrow \langle l_j, \text{std} \rangle \\
I6 & \langle (l_j, l_{j+1}), \text{dec} \rangle & \Rightarrow \langle l_j, \text{dec} \rangle \\
I7 & \langle (l_j, l_{j+1}), \text{dec} \rangle & \Rightarrow \langle (l_j, l_{j+1}), \text{dec} \rangle \\
I8 & \langle (l_j, l_{j+1}), \text{inc} \rangle & \Rightarrow \langle l^*, \text{std} \rangle \\
I9 & \langle (l_j, l_{j+1}), \text{dec} \rangle & \Rightarrow \langle l^*, \text{std} \rangle
\end{tabular}

```

其中 l^* 是新界标值， $l_j < l^* < l_{j+1}$ 。

4.5.2 QSIM 算法

QSIM 算法可对系统的行为进行定性仿真。首先将初始状态送入 ACTIVE 表中，然后重复 (1) — (6) 直至 ACTIVE 表空为止。

算法 4.1 QSIM 算法。

- (1) 从 ACTIVE 表中选一状态。
- (2) 对每个参数按转换表找出所有可能的转换。
- (3) 对约束中变元的转换生成二元组, 三元组集合, 依约束关系做一致性滤波。
- (4) 对有公共变元的约束, 对元组进行组对, 再对组对的元组做一致性滤波。
- (5) 从剩下的元组生成所有可能的全局解释。每个解释生成一个新状态作为当前状态的后继状态。

(6) 对新状态做全局滤波, 剩下的状态送入 ACTIVE 表。全局滤波排除下列状态:

- 无变化情形: 如 I1, I4, I7;
- 循环情形: 新状态与某个前辈状态相同;
- 发散情形: 某参数值为 ∞ , 这时当前时间点必为结束点。

这里以上抛球运动过程的定性模拟为例来说明 QSIM 算法。设球的高度为 V , 速度为 I , 加速度为 A 。

已知约束关系为: DERIV(V, I)

DERIV(I, A)

$A(t) = g < 0$

初状态 (t_0, t_1) 球向上运动:

$QS(A, t_0, t_1) = \langle g, std \rangle$

$QS(V, t_0, t_1) = \langle (0, \infty), dec \rangle$

$QS(I, t_0, t_1) = \langle (0, \infty), inc \rangle$

对每个参数做各种可能的转换, 当前处于时间区间上, 所以需使用 I 转换:

A	I1	$\langle g, std \rangle$	\Rightarrow	$\langle g, std \rangle$
V	I5	$\langle (0, \infty), dec \rangle$	\Rightarrow	$\langle 0, std \rangle$
	I6	$\langle (0, \infty), dec \rangle$	\Rightarrow	$\langle 0, dec \rangle$
	I7	$\langle (0, \infty), dec \rangle$	\Rightarrow	$\langle (0, \infty), dec \rangle$
	I9	$\langle (0, \infty), dec \rangle$	\Rightarrow	$\langle I^*, std \rangle$
I	I4	$\langle (0, \infty), inc \rangle$	\Rightarrow	$\langle (0, \infty), inc \rangle$
	I8	$\langle (0, \infty), inc \rangle$	\Rightarrow	$\langle I^*, std \rangle$

下面对约束形成元组集合, 先对单个约束做一致性滤波, 滤掉的以 C 表示, 然后对元组做组对再做一致性滤波, 滤掉的以 W 表示。

根据 DERIV(I, V) 组对:

(I4, I5)	C	(I8, I5)	W
(I4, I6)	C	(I8, I6)	
(I4, I7)		(I8, I7)	C
(I4, I9)	W	(I8, I9)	C

根据 DERIV(V, A) 组对:

(I5, I1)	C	(I7, I1)	
(I6, I1)		(I9, I1)	C

其中如元组 (I4, I5) 中, I4 使 V 的定性状态变为 $\langle (0, \infty), inc \rangle$, 而 I5 使 V 的定性状态变为 $\langle 0, std \rangle$, 这与约束 DERIV(V, I) 不一致, 于是 (I4, I5) 被过滤掉了。又元组 (I4, I9) 中的 I9 和元组 (I9, I1) 中的 I9 都是对 V 的转换, 由于 (I9, I1) 已被过滤掉了, 所以导致 (I4, I9) 被滤掉。

对剩下的元组形成两个全局解释如下：

Y	V	A
I4	I7	I1
I8	I6	I1

第一个解释为无变化，被滤掉。第二个解释是唯一的后继状态。这时

$$QS(A, t_i) = \langle g, \text{std} \rangle$$

$$QS(V, t_i) = \langle 0, \text{dec} \rangle$$

$$QS(Y, t_i) = \langle Y_{\max}, \text{std} \rangle$$

其中 Y_{\max} 是新的界标值。

4.6 代数方法

Williams 建立了一个定性定量相结合的混合代数，实现了相应的符号代数程序 MINIMA，它是 MACSYMA 的一种定性模拟，为化简、分解、组合定性方程提供了工具。这个代数系数用来解决一类物理问题的设计。

这个代数系统是定义在实数 R 和符号 $S' = \{+, -, 0, ?\}$ 之上的。允许在 R 上做定量运算，也允许在 S' 上做定性运算。如可在 R 上运行 $+$, $-$, \times , $/$ ，可在 S' 上进行 \oplus , \ominus , \otimes , \oslash ，并有定性算子 $[]$ 。交换律、结合律、分配律都成立，只是 $[S', \oplus, \otimes]$ 上对 \oplus 来说没有逆元。从而：

$$\text{由 } s \oplus u = t \oplus u \quad \text{导不出 } s = t$$

$$\text{由 } s \oplus t = u \quad \text{导不出 } s = u \ominus t$$

这个系统可用来进行设计。例如，已知一个自行打取的饮料瓶和一个饮料储存箱。要求设计一个装置以便能自动改变瓶箱液面高度，使得当瓶液面高 H 下降时可从箱中得到饮料的补充。这个设计过程可这样来直观地设想：

瓶中液面高 H_b 的升降由流入流出的饮料流量 Q_b 所决定，容器底部压力 P 和饮料密度成比例，也即压力由高度决定。要求瓶中压力相对箱中压力下降时，就有饮料由箱中流入瓶中。显然所需设计的装置，只需由瓶箱间加一条管子来实现。这个设计的推理过程不只涉及具体数值，也不只涉及定性的符号值，有些地方需要精确的关系而不仅是简单的符号关系。

可用所述的混合代数来描述推演这个问题，MINIMA 系统可自动处理这个设计问题：

$$\text{目标 } H_v - H_b = \left[\frac{d}{dt} H_b \right]$$

$$H_b \times A_b = V_b \quad \text{容器模型，瓶中饮料体积为截面乘以高}$$

$$H_b = V_b / A_b$$

$$H_v - H_b = \left[\frac{d}{dt} \left(\frac{V_b}{A_b} \right) \right]$$

$$H_v - H_b = \left[\left(\frac{d}{dt} V_b \right) / A_b \right]$$

$$H_v - H_b = \left[\left(\frac{d}{dt} V_b \right) \right] \oslash [A_b]$$

$$[A_b] = [+]$$

容器模型

$$[H_v - H_b] = \left[\frac{d}{dt} V_b \right]$$

$$Q_b = \frac{d}{dt} V_b$$

容器模型

$$[H_v - H_b] = [Q_b]$$

$$P_v = d \times g \times H_v$$

容器模型，箱中饮料一点的压力是密度与重力加速度、高度的乘积

$$H_v = P_v / (d \times g)$$

$$[P_v / (d \times g) - H_b] = [Q_b]$$

$$P_b = d \times g \times H_b$$

容器模型

$$H_b = P_b / (d \times g)$$

$$[P_v / (d \times g) - P_b / (d \times g)] = [Q_b]$$

$$[P_v - P_b] \ominus ([d] \times [g]) = [Q_b]$$

$$[d] = [+]$$

饮料性质

$$[g] = [+]$$

重力性质

$$[P_v - P_b] \oslash ([+] \otimes [+]) = [Q_b]$$

$$[P_v - P_b] = [Q_b]$$

最后这个表达式，正是一根管子两端压力与流量间的关系。从而只需用一根管子把饮料瓶与饮料箱连接起来。

4.7 几何空间定性推理

空间定性推理是对几何形状或者运动性质进行定性推理，首先需对空间位置及运动方式进行定性表示，进而对几何形状及运动性质进行推理研究及预测分析，并作出逻辑解释。空间定性推理是通过定义一组空间并寻找这些关系间的联系来进行的。目前主要的研究是针对空间定性建模方式、空间形状及关系的定性表示和定性技术的形式化等，产生解释理论，但总体来看与解决工程问题距离尚远。目前，将 Allen 的时态逻辑[Allen 1984]与 Randall 的空间逻辑[Randall 1992]结合起来，形成空间、时间、连续运动的表达逻辑。

另外从空间定性推理派生出空间规划理论，可用于为一组几何对象寻找满足一组约束的分布设计，有关方法主要用于设计自动化、定性建模等领域。在这一领域已取得了一些较有实际意义的成果，如约束满足问题(CSP)求解理论，而实际上很多空间定性规划都是一个几何约束满足问题(GCSP)。

4.7.1 空间逻辑

对空间几何形体及其运动进行几何仿真的主要任务是对其可能状态进行展望(Envisionment)。所谓“展望”是指对系统建模并产生其可能状态的生成树。展望可分为两类，完整的(total)与可达的(attainable)。可达性展望是对建模系统从某一些特殊状态开始建立其可能状态生成树；而完整性的展望可以产生系统的全部可能状态[Cui 1992]。

1992年由Randell等人建立起来的RCC空间时间逻辑是用于对空间问题进行可达性的展

望，并已程序实现。与 Kuipers 的 QSIM 方法类似，基于 RCC 逻辑的仿真算法也是从对系统进行结构性的描述开始的，系统将初始状态作为生成树的根结点，可能的行为则是树中从根结点到叶结点的路径。

空间逻辑的基础在于假设一个原语性的二元关系 $C(x, y)$ 。其中 x, y 表示两个区域 (region)，谓词 C 表示共享一个以上公共点。也就是指相接触，它具有自反性、对称性。

1. 八个基本关系的定义

使用关系 $C(x, y)$ ，一组基本的二元关系可以被定义为：

- (1) $DC(x, y)$ ：表示两区域不相接触。
- (2) $EC(x, y)$ ：表示两区域外部接触。
- (3) $PO(x, y)$ ：表示两区域部分覆盖。
- (4) $=(x, y)$ ：表示两区域完全相同。
- (5) $TPP(x, y)$ ：表示 x 是 y 的一个严格部分并且 x, y 相切 (内切)。
- (6) $NTPP(x, y)$ ：表示 x 是 y 的一个严格部分但 x, y 不相接触 (包含而不相切)。
- (7) $TPP(y, x)$ ：表示 y 是 x 的一个严格部分并且 x, y 相切。
- (8) $NTPP(y, x)$ ：表示 y 是 x 的一个严格部分但 x, y 不相接触。

这八个关系的定义可以通过 $C(x, y)$ 以及一些辅助性的描述函数来构造，其中使用了一些过渡性的状态谓词如 $P(x, y)$ (表示部分属于)， $PP(x, y)$ (表示严格部分属于)， $O(x, y)$ (表示覆盖) 等。

2. 基本关系间的联系

这种空间逻辑与 Allen 的逻辑相类似，也使用预计算的传递性表来表示二元关系之间可能的变化联系，在表中从任一关系 $R_3(a, c)$ 可查找出所有可能的二元关系 $R_1(a, b)$ 与 $R_2(b, c)$ 。这一表对定性仿真是很有用的。然而，近年的研究中尚未给出这类传递性表的建立算法。但 Randell 提到在其仿真程序中使用了该表来检验展望过程中状态描述的一致性。

另外还需提到，几何体与几何区域间的联系用函数 $space(x, t)$ 表示，它表示几何体 x 在 t 时刻占据的几何区域为 $space(x, t)$ ，不考虑时间时，参数 t 被略去，通常在考虑问题中，为了简化运算，可在不产生歧义时直接用变量 x 表示几何体 x 所占据的几何区域。

3. 基本状态间的相互转换

根据两个区域的形状不同，上述八个基本关系可被分为 6 个子集：

- (1) $DC \quad EC \quad PO \quad =$
- (2) $DC \quad EC \quad PO \quad TPP$
- (3) $DC \quad EC \quad PO \quad TPP^{-1}$
- (4) $DC \quad EC \quad PO \quad TPP \quad NTPP$
- (5) $DC \quad EC \quad PO \quad TPP^{-1} \quad NTPP^{-1}$
- (6) $DC \quad EC \quad PO$

这六个子集的划分可以这样理解：如果两个几何区域形状完全相同，那么它们在空间存在，只能是第 (1) 子集所列举的四种情况。而如果有半径相同的一个球体与一个半球体，那么它们之间的关系只能是第 (2) 子集所列举的情况。一个半径为 R 的圆盘和一个半径为 $R/2$ 的圆柱空间就只能是第 (6) 子集的三种情况。

值得注意的是第 (3) 子集、第 (5) 子集分别是第 (2)、(4) 子集的反集，即如果 x, y 是两个空间区域，它们的形状决定它们之间可能的关系 $R(x, y)$ 所构成的集合就是第 (2) 子集，那么当我们考虑关系 $R(x, y)$ 时，所得到的集合就是第 (3) 子集。所以我们可以这样认为：从 (1) 到 (2)，(3) 再到 (4)，(5) 最后到 (6) 是两个几何区域的形状之间的关系从特殊向一般变化的一种体现。

这样就可以得到两个形状不变的区域在空间运动而使它们之间的关系产生变化，其变化

只限于以下四种序列：

- (1) $DC \leftrightarrow EC \leftrightarrow PO \leftrightarrow =$
- (2) $DC \leftrightarrow EC \leftrightarrow PO \leftrightarrow TPP(TPP^{-1})$
- (3) $DC \leftrightarrow EC \leftrightarrow PO \leftrightarrow TPP(TPP) \leftrightarrow NTPP(NTPP^{-1})$
- (4) $DC \leftrightarrow EC \leftrightarrow PO$

4.7.2 空间时间关系描述

1. 方位性状态和运动性状态

Galton 将八个 RCC 关系分为方位性状态(position state)和运动性状态(motion state)，并且使用 Allen 关于时间关系的一些逻辑化的形式，给出了这种分类的定义。首先介绍一下 Galton 使用的概念、谓词和函数。

- (1) 对时间的描述分为区间和时刻；
- (2) 状态存在的描述谓词：
 $\text{Holds-on}(s, i)$ 表示在区间 i 上存在状态 s ；
 $\text{Holds-at}(s, t)$ 表示在时刻 t 存在状态 s ；
- (3) 谓词 $\text{Div}(t, i)$ 表示时刻 t 在区间 i 上；
- (4) 函数 $\text{inf}(i)$ 表示在区间 i 的开始时刻；
- (5) 函数 $\text{sup}(i)$ 表示在区间 i 的结束时刻；

定义 4.4 方位性状态。如果状态 s 满足

$$\forall i (\text{Holds-on}(s, i) \rightarrow \text{Holds-at}(s, \text{inf}(i)) \wedge \text{Holds-at}(s, \text{sup}(i)))$$

即如果一个状态 s 在时间区间 i 上存在，则在该区间的起止时刻这个状态都存在。具有这一性质的状态称为方位性状态。

定义 4.5 运动性状态。如果状态 s 满足

$$\forall t (\text{Holds-on}(s, t) \rightarrow \exists i (\text{Div}(t, i) \wedge \text{Holds-on}(s, i)))$$

即如果在某时刻 t 有状态 s ，那么一定存在包含这一时刻的某一区间使 s 在整个区间都存在，具有这一性质的状态称为运动性状态。

这样，前面八个基本关系可分为两类：

方位性状态： $EC, =, TPP, TPP^{-1}$

运动性状态： $DC, PO, NTPP, NTPP^{-1}$

这两类状态可以通过它们的表现形式加以区分：方位性状态是有“临界”的性质，而运动性状态具有“稳定”的性质。

2. 扰动原理

Galton 根据这种分类给出了扰动原理，这是对空间状态在时域变化进行描述的一组公理体系：

定义 4.5 扰动(perturbation)。如果 RCC 关系 R 与 R' 满足条件：

$$\exists t (\text{Holds-at}(R(a, b), t) \wedge (\exists i (\text{Holds-on}(R'(a, b), i) \wedge (\text{inf}(i)=t \vee (\text{sup}(i)=t))))))$$

即如果时刻 t 有状态 R , 且有一个区间 i 开始或结束于 t 区间 i 上的状态为 R' 。此时称 R 与 R' 互为扰动。

扰动原理: 每个 RCC 关系是它自身的扰动, 另外一个静止性的状态只能与一个运动性的状态互扰动, 反之亦然(只讨论刚体)。

设 R 为一个 RCC 状态, R_1, R_2, \dots, R_n 为 R 的所有扰动, 然后根据扰动原理有以下 6 条公理:

$$(A1) \text{ Holds-on}(R(a, b), i) \rightarrow \bigvee_{i=1}^n \text{ Holds-at}(R_i(a, b), \sup(i));$$

$$(A2) \text{ Holds-on}(R(a, b), i) \rightarrow \bigvee_{i=1}^n \text{ Holds-at}(R_i(a, b), \inf(i));$$

$$(A3) \text{ Holds-at}(R(a, b), t) \rightarrow \exists t' \bigvee_{i=1}^n \text{ Holds-on}(R_i(a, b), (t, t'));$$

$$(A4) \text{ Holds-at}(R(a, b), t) \rightarrow \exists t' \bigvee_{i=1}^n \text{ Holds-on}(R_i(a, b), (t', t));$$

$$(A5) \text{ Holds-on}(s, (t_1, t_2)) \wedge \neg \text{Holds-at}(s, t_3) \wedge t_2 < t_3 \rightarrow \exists t \text{ Holds-on}(s, (t_1, t)) \wedge$$

$$\forall t' (t < t' \rightarrow \neg \text{Holds-on}(s, (t, t')));$$

$$(A6) \text{ Holds-on}(s, (t_2, t_3)) \wedge \neg \text{Holds-at}(s, t_1) \wedge t_1 < t_2 \rightarrow \exists t \text{ Holds-on}(s, (t_1, t_2)) \wedge \forall t'$$

$$(t < t' \rightarrow \neg \text{Holds-on}(s, (t', t_2))).$$

公理(A1), (A2), 表明如果在时间区间 i 上有关系 R , 则在这一区间的起止时刻必有一个 R 的扰动关系存在。

公理(A3), (A4)表明, 如果 t 时刻有关系 R 则必存在 t'_1 与 t'_2 时刻, 使 (t'_1, t) 与 (t, t'_2) 上分别有一个 R 的扰动关系存在。

公理(A5), (A6)表明如果在区间 (t_1, t_2) 上有状态 s , 而 t_3 时刻不再有, 则在 (t_2, t_3) ($t_3 > t_2$) 或 (t_3, t_1) ($t_3 < t_1$) 上必有一时刻 t 使 s 状态发生突变。

4.7.3 空间和时间逻辑的应用

上述公理系统可用于对空间形体之间关系的推理。例如, 当知道在 t_1 时刻有 $DC(a, b)$, 而在 t_2 时刻观察到 $PO(a, b)$, 且 $t_1 < t_2$, 则必可推得在 (t_1, t_2) 上有 t , $EC(a, b)$, 即

$$\text{Holds-at}(DC(a, b), t_1) \wedge \text{Holds-at}(PO(a, b), t_2) \wedge t_1 < t_2 \rightarrow$$

$$\exists t (\text{div}(t, (t_1, t_2)) \wedge \text{Holds-at}(EC(a, b), t))$$

这些理论也可用于进行事件的描述与推理。引入谓词:

$\text{Occurs-at}(T, t)$ 表示事件 T 发生于时刻 t ;

$\text{Occurs-on}(T, i)$ 表示事件 T 发生于时刻 i ;

函数 $\text{Trans}(R_1, R_2)$ 表示状态 R_1 变为 R_2 这样一个事件。

两个谓词将事件与时间相联系, 函数 Trans 在状态与事件之间建立了映射关系。Galton 将事件分为 7 种:

(1) R_1 为方位性状态, R_2 为运动性状态, R_2 为 R_1 的扰动;

(2) R_1 为运动性状态, R_2 为方位性状态, R_2 为 R_1 的扰动;

- (3) R_1 与 R_2 是有共同扰动 R_3 的运动性状态;
- (4) R_1 与 R_2 是无共同扰动的运动性状态;
- (5) R_1 与 R_2 都是方位性状态;
- (6) R_1 为方位性状态, R_2 为运动性状态, 但互不为扰动;
- (7) R_1 为运动性状态, R_2 为方位性状态, 但互不为扰动。

每种情况事件发生的条件都可以用以上介绍的逻辑加以描述。实际上这 7 种情况中第 (1) (2) 种只能是瞬时性的, 即发生于某一时刻, 第 (4) (5) (6) (7) 只能是持续性的, 即发生于某一区间, 而第 (3) 种情况即可能是瞬时性的又可能是区间性的。关于事件的产生与条件, 在 [Galton, 1993] 中给出了详细的讨论。

4.7.4 Randell 算法

Randell 的仿真程序从初始状态开始, 根据约束和规则产生所有可能的状态生成树, 然后给出系统的行为描述、预测和解释 [Cui 1992]。这一方法与 Kuipers 的 QSIM 是类似的。约束可分为两类, 状态内的 (intastate) 与状态间的 (interstate)。例如一个变形虫 (amoeba) 摄取了一个食物, 那食物成了变形虫的一部分, 那么这一状态将保持, 这就是一个状态间约束。状态内约束直接用子句 Φ 表示。状态间约束具有下列形式:

$$\Phi(R_o \Rightarrow (R_1 \vee R_2 \dots \vee R_n))$$

或

$$\Phi(R_o \nRightarrow (R_1 \vee R_2 \dots \vee R_n))$$

上述两式分别表示, 状态 R_o 如果发生, 后继状态将是 $(R_1 \vee R_2 \dots \vee R_n)$, 或者, 后继状态将一定不会是 $(R_1 \vee R_2 \dots \vee R_n)$ 。

Randell 在其仿真程序中还引入添加与删除规则, 添加规则为系统下一状态在区域中引入一个物体, 而删除规则反之。例如, 一旦变形虫摄取食物将产生一个液泡, 而一旦液泡中充满废物被排出体外后将被删除。

(1) 添加规则的表达式如下:

$$\text{add } O_1, O_2, \dots, O_n \text{ with } \psi_1 \text{ when } \psi_2$$

即当 ψ_2 成立时, 系统中增加 O_1, O_2, \dots, O_n , 并有 ψ_1 。

(2) 删除规则的表达式如下:

$$\text{delet } O_1, O_2, \dots, O_n \text{ when } \psi_2$$

表示 ψ_2 成立时, 删除 O_1, O_2, \dots, O_n 。

算法 4.2 Randell 算法。假设初始状态 S_0 已放入状态集合 S 中。

- (1) 如果 S 空则停止;
- (2) 从 S 中选出状态 S_i , 且移出;
- (3) 如果 S_i 为不一致状态将转 (2);

- (4) 应用状态约束选择可用变换规则;
- (5) 用所选出的规则产生可能的下一个状态集合;
- (6) 使用添加与删除规则;
- (7) 进行状态内约束检查;
- (8) 将剩下的状态加入 S 并转 (1)。

习 题

- 1. 定性的含义是什么，什么是定性推理。
- 2. 简述几种基本的定性推理方法。
- 3. 定性推理方法要解决怎样的问题，描述它的分析步骤。
- 4. 请用定性进程方法描述锅炉加热的过程。
- 5. 定性仿真推理是一种比较重要的定性推理架构，简要叙述它的基本方法。
- 6. 试比较基于代数方法和基于几何空间的定性推理方法。
- 7. 定性推理已经在经济分析预测中实际使用，请查阅资料学习一个定性推理的系统。

第五章 基于范例的推理

5.1 概述

人们为了解决一个新问题，先是进行回忆，从记忆中找到一个与新问题相似的范例，然后把该范例中的有关信息和知识复用到新问题的求解之中。以医生看病为例，在他对某个病人做了各种检查之后，会想到以前看过的病人情况。找出在几个重要症状上相似的病人，参考那些病人的诊断和治疗方案，用于眼前的这个病人。

在基于范例推理 (Case-Based Reasoning, 简称 CBR) 中，把当前所面临的问题或情况称为目标范例(target case)，而把记忆的问题或情况称为源范例(base case)。粗略地说，基于范例推理就是由目标范例的提示而获得记忆中的源范例，并由源范例来指导目标范例求解的一种策略。

Kolodner 在《Case-Based Reasoning》中对范例(case)给出了一个定义：“范例是一段带有上下文信息的知识，该知识表达了推理机在达到其目标的过程中能起关键作用的经验”[Kolodner 1993]。具体来说，一个范例应具有如下特性：

- 范例表示了与某个上下文有关的具体知识，这种知识具有可操作性。
- 范例可以是各式各样的，可有不同的形状和粒度，可涵盖或大或小的时间片，可带有问题的解答或动作执行后的效应。
- 范例记录了有用的经验，这种经验能帮助推理机在未来更容易地达到目标，或提醒推理机失败发生的可能性有多大等等。

基于范例推理是人工智能发展较为成熟的一个分枝。它是一种基于过去的实际经验或经历的推理。传统的推理观点把推理理解为通过前因结果链（如规则链）导出结论的一个过程。许多专家系统使用的就是这种规则链的推理方法。基于范例推理则是另一种不同的观点。它使用的主要知识不是规则而是范例，这些范例记录了过去发生的种种相关情节。对基于范例推理来讲，求解一个问题的结论不是通过链式推理产生的，而是从记忆里或范例库中找到与当前问题最相关的范例，然后对该范例作必要的改动以适合当前问题。

基于范例推理作为一种方法论是合理的。因为客观世界有两个特点：规整性和重现性。世界从总体上看存在一定的规整性，相似条件下发生的动作会产生相似的结果。“历史是惊人的相似”，过去的经历很有可能预示未来。

基于范例推理的研究起源于从认知科学的角度对人类的推理和学习机制进行探索[136]。从小孩的简单活动到专家的慎重决策，人类借助于有意识地或无意识地回忆完成各种事务。人类经常是按经验行事，而人又是某种意义上的一个智能系统，因此自然可以把这种基于经验的推理方法用于人工智能的研究和应用上。总体上说，基于范例推理在如下方面对人工智能作出了贡献：

(1)知识获取：这是基于知识的系统的瓶颈问题。开发基于规则的知识系统时，获取规则或模型是最繁琐的一件事务，需要领域专家和知识工程师的密切合作；有的领域甚至很难找到适合的规则。

(2)知识维护：随着系统的运行，知识系统常常出现初始的知识不完整而需要更新，新的知识可能会与原有知识产生冲突，导致非常大的系统变动。基于范例推理则不存在这些问题。

(3)改进问题求解效率：基于范例推理通过复用过去的解答，无需象常规推理那样从头做起。特别是，由于记录了过去求解时的失败或成功信息，使得求解新问题时可避开错误的途径。

(4)改进问题求解质量：过去求解失败的经历可以指导当前求解时避开失败。

(5)提高用户接受度：用户如果能清楚知道系统得出的结论是合理地推出的，他才相信该结论。基于范例推理的根据则是历史事实，事实胜于雄辩，因此对用户有说服力。

中国科学院计算技术研究所智能信息处理开放实验室在基于范例推理方面进行了一系列研究。1991年史忠植、李宝东提出了记忆网模型和范例检索算法[李宝东 1991]。1993年周涵研制了基于范例学习的内燃机油产品设计系统EOFDS[周涵 1993]。1994年徐众会开发了基于范例推理的天气预报系统。1996年王军开发了基于范例推理的淮河王家坝洪水预报调度系统FOREZ。2000年研制了渔情分析专家系统[叶施仁 2001]

5.2 类比的形式定义

用类比求解问题，往往在提出或遇到某一问题时，回忆以前相似的老问题，通过对两种情况进行匹配，经过推理获得新知识。也可以通过对老问题解法的检索和分析、调整，得出新问题的解决方法。因此，计算模型除了记忆和新问题相似的老问题的解法外，还应具有获取技能的过程，即必须学会根据过去有用的经验，来调整问题求解方法。当人们对存在相似解进行更为直接的回忆和修改后仍不能得出问题的解答时，再反过来用弱方法求解。因此，类比学习是一种基于知识(或经验)的学习。类比求解问题的一般模式如图5.1所示。

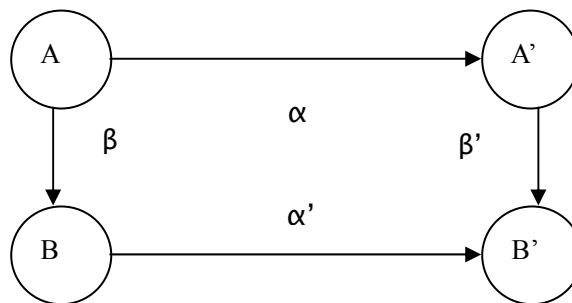


图5.1 类比求解问题的一般模式

类比问题求解的形式可描述为：已知问题A，有求解结果B，现给定一个新问题A'，A'与A在特定的度量下是相似的，求出问题A'的求解结果B'。如图5.1所示， β 反映B与A之间的依赖关系，称作因果关系。 α 表示源领域(source domain)A与目标领域(target domain)A'之间的相似关系。由此可以推出，B'与A'之间的依赖关系 β' 。下面我们给出有关类比学习的一些定义。

定义5.1 相似性。 P_1, P_2 是谓词的有限集，谓词名 $q_1 \in P_1, q_2 \in P_2$ 是相同谓词，则对pair $\langle q_1, q_2 \rangle$ 在 $P_1 \times P_2$ 中存在相似关系。

定义5.2 部分匹配。 $s \in S, t \in T, s$ 和 t 是包括公共常数的元文字的有限集。对于 Q ，设 $Q\theta \subseteq s \subseteq s \times t, Q\theta$ 和 $v(Q)\theta$ 存在一对一的对应关系，则 (Q, θ) 称作对 s 和 t 一般部分匹配。

定义5.3 大小程度。 设 $(Q, \theta), (Q', \theta')$ 是对于 $s \times t$ 部分匹配。如果存在替换 ξ ，使 $Q'\xi \subseteq Q$ ，对于任何 $w \in v(Q') w\theta' = w\xi\theta, (Q, \theta)$ 是大于 (Q', θ') ，可以写成 $(Q, \theta) \geq (Q', \theta')$ 。

定义5.4 最大部分匹配。 在 $s \times t$ 中，对于 $(Q, \theta), (Q', \theta')$ 部分匹配，如果 $(Q, \theta) \geq (Q', \theta')$ ，那么 (Q, θ) 称作在 $s \times t$ 中的最大部分匹配。

定义 5.5 类比学习。 对于 $s_1, s_2 \in S, t_1 \in T$ 和 β 在 $S \times S$ 中， $s_1 \times s_2 \in \beta, m$ 在 $s_1 \times t_1$ 是最大部分匹配。根据 $t_1 \times t_2 \in \beta$ 和 m 在 $s_2 \times t_2$ 中是最大部分匹配，将得到 $t_2 \in T$ ，这就是类比学习。

5.3 过程模型

基于范例推理是类比推理的一种。在基于范例推理中，最初是由于目标范例的某些（或者某个）特殊性质使我们能够联想到记忆中的源范例。但它是粗糙的，不一定正确。在最初的检索结束后，我们须证实它们之间的可类比性，这使得我们进一步地检索两个类似体的更多的细节，探索它们之间的更进一步的可类比性和差异。在这一阶段，事实上，已经初步进行了一些类比映射的工作，只是映射是局部的、不完整的。这个过程结束后，获得的源范例集已经按与目标范例的可类比程度进行了优先级排序。接下来，我们便进入了类比映射阶段。我们从源范例集中选择最优的一个源范例，建立它与目标范例之间的一致的一一对应。下一步，我们利用一一对应关系转换源范例的完整的（或部分的）求解方案，从而获得目标范例的完整的（或部分的）求解方案。若目标范例得到部分解答，则把解答的结果加到目标范例的初始描述中，从头开始整个类比过程。若所获得的目标范例的求解方案未能给目标范例以正确的解答，则需解释方案失败的原因，且调用修补过程来修改所获得的方案。系统应该记录失败的原因，以避免以后再出现同样的错误。最后，类比求解的有效性应该得到评价。整个类比过程是递增地进行的。图5.2 给出了基于范例推理的一般结构。

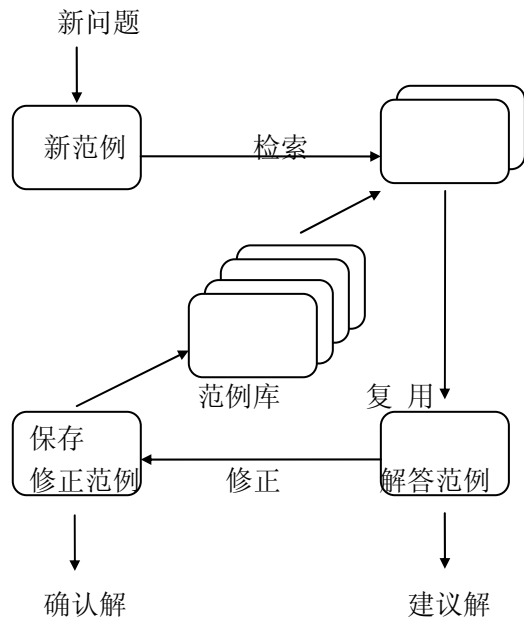


图 5.2 基于范例推理结构示意图

基于范例推理有两种形式：问题求解型（problem-solving CBR）和解释型（interpretive CBR）。前者利用范例以给出问题的解答；后者把范例用作辩护的证据。范例推理的过程见图 5.3。在范例推理中，关心的主要问题如下：

- (1) 范例表示：基于范例推理方法的效率和范例表示紧密相关。范例表示涉及这样几个问题：选择什么信息存放在一个范例中；如何选择合适的范例内容描述结构；范例库如何组织和索引。对于那些数量达到成千上万、而且十分复杂的范例，组织和索引问题尤其重要。
- (2) 分析模型：分析模型用于分析目标范例，从中识别和抽取检索源范例库的信息。
- (3) 范例检索：利用检索信息从源范例库中检索并选择潜在可用的源范例。基于范例推理方法和人类解决问题的方式很相近。碰到一个新问题时，首先是从记忆或范例库中回忆出与当前问题相关的最佳范例。后面所有工作能否发挥出应有的作用，很大程度上依赖于这一阶段得到的范例质量的高低，因此这步非常关键。一般讲，范例匹配不是精确的，只能是部分匹配或近似匹配。因此，它要求有一个相似度的评价标准。该标准定义得好，会使得检索

出的范例十分有用，否则将会严重影响后面的过程。

(4) 类比映射：寻找目标范例同源范例之间的对应关系。

(5) 类比转换：转换源范例中同目标范例相关的信息，以便应用于目标范例的求解过程中。其中，涉及到对源范例的求解方案的修改。把检索到的源范例的解答复用于新问题或新范例之中。它们分别是，源范例与目标范例间有何不同之处；源范例中的哪些部分可以用于目标范例。对于简单的分类问题，仅需要把源范例的分类结果直接用于目标范例。它无需考虑它们之间的差别，因为实际上范例检索已经完成了这项工作。而对于问题求解之类的问题，则需要根据它们之间的不同对复用的解进行调整。

从复用的信息内容来看，主要有两种类型：结果的复用和方法的复用。对于前者来讲，当源范例的解答结果需要调整时，它依据一些转换操作知识，把源范例中的种种可能解转换为目标范例中相应的解。方法的复用则关心源范例中的问题是如何求解的，而不是其解答结果。源范例带有求解方法的信息，如操作算子的使用，子目标的考虑，成功或失败的搜索路径等。复用时需把这些方法重新例化。

(6) 解释过程：对把转换过的源范例的求解方案应用到目标范例时所出现的失败做出解释，给出失败的因果分析报告。有时对成功也同样做出解释。基于解释的索引也是一种重要的方法。

(7) 范例修补：有些类似于类比转换，区别在于修补过程的输入是解方案和一个失败报告，而且也许还包含一个解释，然后修改这个解以排除失败的因素。

当复用阶段产生的求解结果不好时，需要对其进行修补。修补的第一步是对复用结果进行评估，如果成功，则不必修补，否则需对错误采取修补。

进行结果评估，可以依据它在实际环境中运行后的反馈，也可以通过咨询完成。等待反馈可能要花一段时间，比如病人治疗的结果好坏。为此，可以考虑通过模拟时间环境来实现。

修正错误一般涉及错误探测和寻找原因。寻找原因是为了对错误进行解释分析，以找出原因对症下药，即修改造成错误的原因使其不再发生。当然，修改既可以使用领域知识模型进行自修补，也可以由用户输入完成。

(8) 类比验证：验证目标范例和源范例进行类比的有效性。

(9) 范例保存：新问题得到了解决，则形成了一个可能用于将来情形与之相似的问题。这时有必要把它加入到范例库中。这是学习也是这是知识获取。此过程涉及选取哪些信息保留，以及如何把新范例有机集成到范例库中。修改和精化源范例库，其中包括泛化和抽象等过程。

在决定选取范例的哪些信息进行保留时，一般要考虑以下几点：和问题有关的特征描述；问题的求解结果；以及解答为什么成功或失败的原因及解释。

把新范例加入到范例库中，需要对它建立有效的索引，这样以后才能对之作出有效的回忆。索引应使得与该范例有关时能回忆得出，与它无关时不应回忆出。为此，可能要对范例库的索引内容甚至结构进行调整，如改变索引的强度或特征权值。

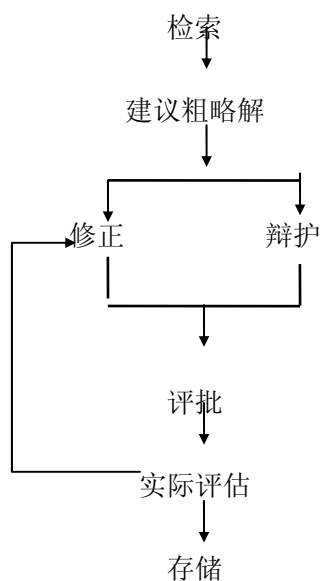


图 5.3 基于范例推理过程示意图

5.4 范例的表示

知识在大脑中的记忆机理现在仍是个悬而未决的问题。虽然在目前的知识系统中使用了产生式、语义网、框架、面向对象等诸多的知识表示方法，但它们在学习系统中，尤其在类比学习系统中却显得有些难于胜任了。原因在于，知识的记忆不仅要使知识成为有结构和有组织的体系，还应保证记忆的知识是易于检索和存取的，而且，还应该是易于学习的。

在生理学、心理学等领域，已经广泛开展了关于记忆的研究。心理学的研究者们注重研究记忆的一般理论，已经提出了许多记忆模型，典型的包括情景记忆 (episodic memory)，语义记忆 (semantic memory)，联想记忆 (associative memory)、Schank 的动态记忆理论 (dynamic memory) 等。

知识是有结构的体系。在某些任务的执行过程中，专家采用语义记忆来存储信息。这种信息记忆方法具有下列优点：

- 有利于检索。
- 易于组织。可以把它们连接成树形层次或者网络。
- 易于管理。知识的改变只对局部产生影响。
- 有利于知识的共享。

Schank 的动态记忆理论把知识记忆在一些结构中。有四种类型的结构, 它们是记忆组织包 (Memory Organization Packet, MOP), 场景 (Scene), 剧本 (Script), 主题记忆包 (Thematic Organization Packet, TOP)。一个记忆组织包中可以包含许多场景，每一场景又可以包含多个剧本。同时，在记忆组织包的上层还可能包含元记忆组织包 (Meta-MOP) 等。这些结构按照一定的组织原则形成一个网络结构，而通过索引来检索它们。

5.4.1 语义记忆单元

语义记忆单元，是指在学习、分析、理解、记忆知识的过程中所着重关注的其中那些概

念、模式、主题等，以及据此形成的关于知识的概念性认识。换言之，这些语义记忆单元是系统对知识经“计算”之后，抽取其中最能反映知识本身特征且可以很好地使知识内在地联系在一起的那些因素而获得的。

我们记忆的知识只有建立在一定程度的加工基础上，才能获得真正的记忆，并很好地服务于以后的使用。语义记忆单元的作用在于，它是对具体知识、具体问题某个方面的概括，以及对具体知识和具体问题的较为抽象的本质的认识。具体知识和具体问题可以依语义记忆单元为中心，以语义记忆单元之间的联系为纽带很好地组织起来。

那么，选择知识中哪些因素作为语义记忆单元呢？由于每种知识都有其自身内在的特点，因此，选择的策略便因知识的特点不同而有所不同。一般地，对于很新的知识，我们往往把其中的概念作为首要的记忆对象。随着关于此类知识的积累愈加丰富，在具备了关于具体问题的分析能力之后，便可用来分析具体问题的主旨，从中概括出一些抽象的概念性的认识。例如，天文学中，天体之间围绕旋转是我们对天体间关系的一般认识，则“围绕旋转”这个二元关系便可以抽象为一个语义记忆单元。通过这个词汇，不仅可以联想到具体的知识，还可以联想到具体的形象。关于选择策略的另外一点考虑是，可以把知识中涉及到的那些重要的模式抽象出来作为语义记忆单元。这些模式并不是用文字表达的，而是用某些特殊符号组合而成的一种特殊的表达方式。

5.4.2 记忆网

我们所记忆的知识彼此之间并不是孤立的，而是通过某种内在的因素相互之间紧密地或松散地有机联系成的一个统一的体系。我们使用记忆网来概括知识的这一特点。一个记忆网便是以语义记忆单元为结点，以语义记忆单元间的各种关系为连接建立起来的网络。在下面的叙述中，我们把语义记忆单元简记为 SMU [Li 1992]。网络上的每一结点表示一语义记忆单元，形式地描述为下列结构：

```
SMU = { SMU_NAME slot
        Constraint slots
        Taxonomy slots
        Causality slots
        Similarity slots
        Partonomy slots
        Case slots
        Theory slots
    }
```

(1) SMU_NAME slot: 简记为 SMU 槽。它是语义记忆单元的概念性描述，通常是一个词汇或者一个短语。

(2) Constraint slots: 简记为 CON 槽。它是对语义记忆单元施加的某些约束。通常，这些约束并不是结构性的，而只是对 SMU 描述本身所加的约束。另外，每一约束都有 CAS 侧面(facet)和THY 侧面与之相连。

(3) Taxonomy slots: 简记为 TAX 槽。它定义了与该 SMU 相关的分类体系中的该 SMU 的一些父类和子类。因此，它描述了网络中结点间的类别关系。

(4) Causality slots: 简记为 CAU 槽。它定义了与该 SMU 有因果联系的其它 SMU，它或者是另一些 SMU 的原因，或者是另外一些 SMU 的结果。因此，它描述了网络中结点间的因果联系。

(5) Similarity slots: 简记为 SIM 槽。它定义了与该 SMU 相似的其它 SMU, 描述网络中结点间的相似关系。

(6) Partonomy slots: 简记为 PAR 槽。它定义了与该 SMU 具有部分整体关系的其它 SMU。

(7) Case slots: 简记为 CAS 槽。它定义了与该 SMU 相关的范例集。

(8) Theory slots: 简记为 THY 槽。它定义了关于该 SMU 的理论知识。

上述 8 类槽可以总地分成三大类。一类反映各 SMU 之间的关系, 包括 TAX 槽、CAU 槽、SIM 槽和 PAR 槽; 第二类反映 SMU 自身的内容和特性, 包括 SMU 槽和 THY 槽; 第三类反映与 SMU 相关的范例信息, 包括 CAS 槽和 CON 槽。关于相似的 SMU, 我们引进一个特殊的结点, 即内涵结点 MMU, 用此来表示与此结点连接的各结点是关于此内涵相似的一些 SMU。通过为 SMU 增加约束, 可以把比 SMU 更为特殊的知识记忆在该 SMU 周围。因此, 通过 SMU 就可以检索到受到一些约束的知识。这使得知识的记忆具有层次性。PAR 槽虽然在我们的模型中并不影响知识的检索, 但却对知识的回忆具有很大的作用。通过部分整体联系, 我们便可以回忆起属于某一主题或者某个领域的知识。THY 槽记忆的是关于 SMU 的理论知识, 如上面给出的“资源冲突”的知识。这些知识可以采用任何成熟的知识表示方法, 例如, 产生式、框架、基于对象的表示方法等。在某些情况下, 这使得知识处理可以局部化。在记忆网中, 结点之间的语义关系保证了同某个 SMU 有关的知识是很容易检索到的。

我们看到, 记忆网是相当复杂的, 但它确实反映了知识之间错综复杂的内在联系。网络的复杂性决定了网络建立和学习的复杂性。对于人来讲, 记忆网是经过知识的长期积累和学习思考的结果而逐步完善和形成的。在这一过程中, 不断地增加新的结点和知识, 同时, 把长久不用的关于某个结点的知识遗忘掉。这说明, 网络建立的过程事实上便是知识的学习过程。

使用记忆网可以一定程度地解释知识的遗忘。一般对于一段时间内不用的知识, 我们往往无意识地把它的一些具体内容遗忘殆尽, 而却能在记忆中留下一些关于这种知识的大致的印象。这说明, 记忆网本身是一种长时性的记忆, 而关于各槽的记忆则是短期的, 会逐渐淡化甚至遗忘。我们可以用记忆强度来描述知识的遗忘和记忆得到强化的现象。一般地, 记忆强度是时间和回忆的一个函数。随着时间的增加, 记忆强度会减弱, 而每一次回忆之后, 获得回忆的知识的记忆强度便有所增加。

记忆网与语义网既有联系, 又有差别, 是在语义网基础上发展起来的一种模型。它们都使用结点来表示信息, 使用结点之间的连接来表示语义关系。它们之间具有很大的不同。对信息的表示是有本质的区别的。语义网的信息表达能力只局限于网络自身, 亦即知识只能通过结点和结点间的连接来表示。但记忆网的表达能力却远不止于此, 表现在:

- 可以记忆使用其它表达方式表示的理论和具体范例。
- 通过为结点施加约束来记忆较为特殊的知识。
- 可以通过内涵结点来组织相似的知识。
- 记忆单元可以是一个主体, 可以独立地完成一定的任务。

在记忆网基础上可以进行多种推理, 例如:

(1) 通过语义关系可以在各结点之间继承知识。这一点类似于语义网的继承推理。

(2) 约束满足是指在结点内部, 通过对其内涵施加约束而获得特殊知识的过程。

(3) 对 THY 槽中所记忆的知识, 可以针对表示方法的不同而采用相应的推理方式, 比如正反向推理和信息传递等。

(4) CAS 槽中记忆了具体的范例, 故可采用基于范例的推理方法。由此, 基于范例的抽象和泛化也是在记忆网上可实施的操作。

5.5 范例的索引

范例组织时由两部分组成，一是范例的内容，范例应该包含哪些有关的东西才能对问题的解决有用；二是范例的索引，它和范例的组织结构以及检索有关，反应了不同范例间的区别。

范例内容一般有如下三个主要组成部分：①问题或情景描述：范例发生时要解决的问题及周围世界的状态；②解决方案：对问题的解决方案；③结果：执行解决方案后导致的结果（周围世界的新的状态）。问题或情景描述和解决方案是必不可少的部分，任何基于范例推理系统必须要有，而结果部分在有的系统中没有。

（1）问题或情景描述 是对要求解的问题或要理解的情景的描述，一般要包括这些内容：当范例发生时推理器的目标，完成该目标所要涉及的任务，周围世界或环境与可能解决方案相关的所有特征。

（2）解决方案 的内容是问题如何在一特定情形下得到解决。它可能是对问题的简单解答，也可能是得出解答的推导过程。

（3）结果 记录了实施解决方案后的结果情况，是失败还是成功。有了结果内容，CBR在给出建议解时有能给出曾经成功地工作的范例，同时也能利用失败的范例来避免可能会发生的问题。当对问题还缺乏足够的了解时，通过在范例的表示上加上结果部分能取得较好的效果。

范例索引对于检索或回忆出相关的有用范例非常重要。索引的目标是：在对已有范例进行索引后，当给定一个新的范例时，如果范例库中有与该范例相关的范例，则可以根据索引找到那些相关的范例。

建立范例索引有三个原则：①索引与具体领域有关。数据库中的索引是通用的，目的仅仅是追求索引能对数据集进行平衡的划分从而使得检索速度最快；而范例索引则要考虑是否有利于将来的范例检索，它决定了针对某个具体的问题哪些范例被复用；②索引应该有一定的抽象或泛化程度，这样才能灵活处理以后可能遇到的各种情景，太具体则不能满足更多的情况；③索引应该有一定的具体性，这样才能在以后被容易地识别出来，太抽象则各个范例之间的差别将被消除。

5.6 范例的检索

范例检索——从范例库(Case Base)中找到一个或多个与当前问题最相似的范例；CBR系统中的知识库不是以前专家系统中的规则库，它是由领域专家以前解决过的一些问题组成。范例库中的每一个范例包括以前问题的一般描述即情景和解法。一个新范例并入范例库时，同时也建立了关于这个范例的主要特征的索引。当接受了一个求解新问题的要求后，CBR利用相似度知识和特征索引从范例库中找出与当前问题相关的最佳范例，由于它所回忆的内容，即所得到的范例质量和数量直接影响着问题的解决效果，所以此项工作比较重要。它通过三个子过程，即特征辨识、初步匹配，最佳选定来实现。

特征辨识是指对问题进行分析，提取有关特征，特征提取方式有：(a)从问题的描述中直接获得问题的特征，如自然语言对问题进行描述并输入系统，系统可以对句子进行关键词提取，这些关键词就是问题的某些特征。(b)对问题经过分析理解后导出的特征，如图象分析理解中涉及的特征提取。(c)根据上下文或知识模型的需要从用户那里通过交互方式获取的特征，系统向用户提问，以缩小检索范围，使检索的范例更加准确。

初步匹配是指从范例库中找到一组与当前问题相关的候选范例。这是通过使用上述特征作为范例库的索引来完成检索的。由于一般不存在完全的精确匹配，所以要对范例之间的特征关系进行相似度估计，它可以是基于上述特征的与领域知识关系不大的表面估计，也可以通过问题进行深入理解和分析后的深层估计，在具体做法上，则可以通过对特征赋予不同的权值体现不同的重要性。相似度评价方法有最近邻法、归纳法等。

最佳选定是指从初步匹配过程中获得的一组候选范例中选取一个或几个与当前问题最相关的范例。这一步和领域知识关系密切。可以由领域知识模型或领域知识工程师对范例进行解释，然后对这些解释进行有效测试和评估，最后依据某种度量标准对候选范例进行排序，得分最高的就成为最佳范例，比如最相关的或解释最合理的范例可选定为最佳范例。

标准的检索过程如图 5.4 的左图所示。此过程的输入为源范例(source case, 即当前范例)，它由当前情景和推理目标构成。通过情景分析，情景描述得到细化；如果在范例库中有与源范例相似的范例，那么源范例与范例库中相似范例相关的索引应该能够通过细化过程计算出来。检索算法使用源范例和细化出来的索引在范例库中搜索。搜索需要借助匹配过程来决定源范例和范例库中遇到的范例之间的匹配度。检索算法返回一组（部分）匹配的相似范例，这些范例都有可能对求解新问题有用。接着还要对这组相似范例进行进一步的分析排位(ranking)，确定最有用的范例。

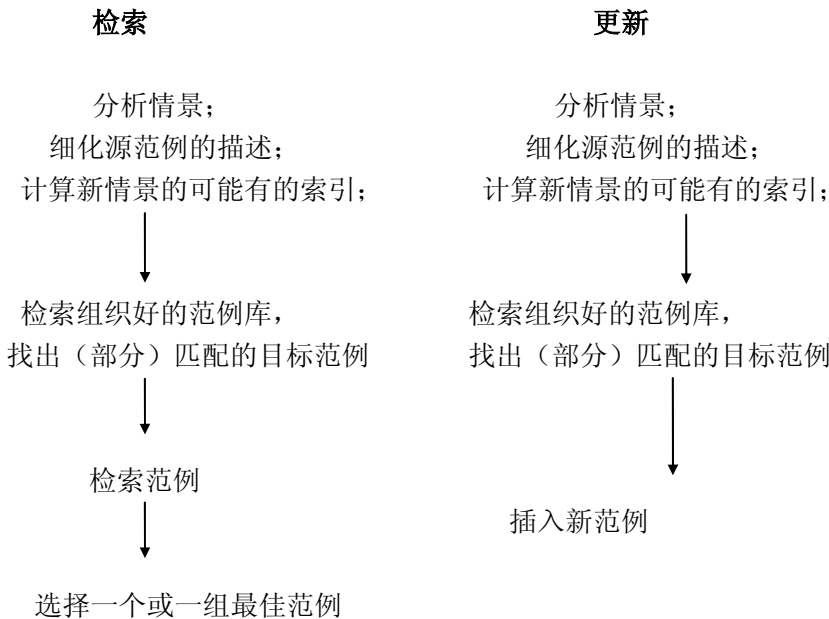


图 5.4 检索和更新过程

检索过程有三个核心部分组成：检索算法、匹配函数，和情景分析(situation assessment)。下面着重讨论检索算法。数据结构和算法之间有着紧密相连的关系，因此探讨检索算法必须和范例库的组织结构联系在一起，不同的范例库组织自然要有相应的不同算法来检索。表结构或平面结构相对较简单，而树和图结构则较复杂。不同的组织形式各有利弊，依照具体情况而定。

CBR 中已形成了一系列的范例组织和检索策略和算法。有串行的、有并行的；有平面型的、有层次型的；有在细粒度级上建立索引以区别不同范例的、有在粗粒度级上建立索引的。用得最多的则是倒排索引之类的方法，它既可以采用串行也可用并行策略来检索。kitano 采用此方法建立的应用系统，已能处理 25,000 多个范例。最常用的检索方法有如下三种：近邻

法、归纳法，以及模板检索。

(1)近邻法：近邻法采用特征间的加权匹配来估计范例之间的相似度。因此此法的关键问题是，如何确定特征的权重。近邻法的缺点是，检索的时间复杂度会随着范例库中范例的个数增多而线性增长。因此，当范例库较小时近邻法是教适合的。

(2)归纳法：采用归纳方法[1]可以确定哪个特征在区分范例时最好，此方法能生成一棵决策树，它可以有效地组织范例。

(3)模板检索：与 SQL 查询类似，模板检索能返回在一定参数值范围内的所有范例。模板法一般用于使用其他技术之前，如在使用近邻法之前，可先用模板法来减小近邻法的搜索空间。

5.7 相似性关系

基于范例的推理很关键的一个环节是范例检索过程得到的相似范例，这是应用成功与否的前提。由于范例检索是在相似比较的基础上进行的，要检索到相似的范例来，完全就要靠“什么程度才算相似”的定义了。如果定义的不好，检索的结果就不理想，也就谈不上应用的成功。因此相似度的定义十分重要。

范例的表示表明，范例的情境是由许多属性组成，范例间的相似度就是根据属性（或变量）之间的相似度定义的。目标范例与源范例之间的相似性有语义相似、结构相似、目标相似和个体相似。

5.7.1 语义相似性

两范例之间是可以类比的，首先必须满足语义上具有相似性关系。相似性关系是类比问题求解的基础。两实体的类比可以区分为正类比、反类比、不确定类比。正类比是由相似性关系所确定的两实体之间的可类比部分，反类比则是已被确定为两实体间不相似的部分，不确定类比是两实体之间尚未确定是否可类比的部分。两个实体可类比的条件之一是：模型的本质性质和因果关系不构成反类比的一部分。不确定类比使得类比具有一定的预见性，这种预见可能是正确的，也可能是错误的。在类比求解中，目标范例的本质特征和源范例的本质特征必须具有相似性关系，才能使类比有了基础。

还有些学者把相似性关系区分为表面的相似性和结构的相似性。表面的相似性定义为对确定问题的解不扮演因果角色的那些共性。相反，影响目标的那些共性被称为结构的相似性。较少限制的定義可以把结构相似性定义为关系之间语义的交叠。两者对类比过程具有重要作用。表面相似性有助于发现最初的类比和辨识个体；结构相似性不仅有助于类比检索，而且对类比映射具有非常大的作用。

5.7.2 结构相似性

如果在两个结构之间存在某种对应关系，且这种对应关系能够保持结构一致性，则认为两结构是同构的。结构一致性要求：一一对应的关系必须保证它们涉及的个体或低阶关系也是一一对应的，且这种对应不应打破原来个体间的对应关系。同构对于类比推理有效性的有重要意义。

结构对于类比检索的意义是重大的。首先，我们发现，表面上并不相似的范例由于在结

构上具有相似性，从而使类比成为可能。原子和太阳系涉及不同的领域，表面上看，并不具有什么本质的联系。然而，深入的研究表明，两者具有十分相似的空间结构。第二，子结构间的同构或相似性可以使我们只需见树木，而不必顾及森林。这是因为，目标范例和源范例的类比有时只是局部的。如果从整体上看，两者可能并不具备任何的相似性。例如，在故事理解方面

，两个故事之间总的来说可能会大相径庭，然而，其中某个情节，或者某个人物的性格等，可能具有惊人的相似之处。在规划方面，我们不仅要考虑整个源方案的可用性，而且，如果尝试使用整个源方案不能得到成功，我们还应该把注意力放在检索子方案上面。有时，放弃整个方案是不明智的。

在我们的类比检索模型中，同构和结构相似性占有非常重要的地位。结构相似性有助于初步检索到可类比的源范例，而同构则提醒我们优先考虑那些与目标问题具有同构和局部同构关系的源范例或者部分源范例。

5.7.3 目标特征

问题求解的最终目的是要实现问题本身所提出的目标。人们求解问题时，都是向着这个目标而竭尽其力。在相似的一组源范例中，那些对实现目标范例的目标具有潜在的重要作用的源范例，较之那些不具有目标相关性的源范例，更应该得到优先考虑。

如果为一种结构表示增加了目标信息，那么，这个增大了的结构，同其它包含有相似的目标信息的结构之间，更加具有语义相似性和结构一致性。换言之，目标特征会增加我们对源范例选择的可靠性。同时，它还可以帮助我们限制对源范例进行搜索的范围。在 Keane 的类比检索模型中，通过目标特征和客体特征来检索可类比的源问题 (Keane 1988)。Keane 把目标特征称为类比检索的“结构线索”。

然而，我们是否对那些不具备目标相关性的源范例弃而不顾了呢？事实上，目标特征只是一个重要的附加约束。如果过分强调目标特征，我们就可能把不具有目标相关性但具有相似的子结构的源范例置于不予考虑的地位。这样做是不明智的。

5.7.4 个体相似性

在我们的模型中强调的另一重要约束是个体的类别信息。从不严格的意义上讲，如果两个个体之间具有一些（或一个）相似的属性，则它们是属于同一类别的。在概念聚类中，我们使用概念（或客体）间的相关性或紧致性来对概念（客体）集进行分类。相关性是指概念的属性之间相似度的平均值。但在这里，我们将把电线和绳索看作是同一类别的，因为它们均可以用来束缚物体。

有时，一个范例中的某些（或某个）个体可能对于问题的解决具有主导作用。在这种情况下，这些（或某个）个体应该作为问题的显著的检索信息来初步地检索源范例。在最初的检索结束之后，对于那些同目标范例的个体具有同类关系或部分和整体的关系的源范例，我们应该给以优先考虑。个体之间的类比有助于我们认识如何使用一个个体。范例的部分解决能帮助发现整个范例的解。

5.7.5 相似性计算

1. 数值性属性的相似度

$$\text{Sim}(V_i, V_j) = 1 - d(V_i, V_j) = 1 - d_{ij}$$

$$\text{或 } \text{Sim}(V_i, V_j) = \frac{1}{1 + d(V_i, V_j)} = \frac{1}{1 + d_{ij}}$$

$$d_{ij} = |V_i - V_j|$$

$$\text{或 } d_{ij} = \frac{|V_i - V_j|}{\max\{V_i, V_j\} - \min\{V_i, V_j\}}$$

(5.1)

其中 V_i, V_j 是某个属性 V 的两个属性值。

2. 枚举型属性的相似度

枚举型属性相似度一般有两种，一种是只要两个属性值不同，就认为两者之间的相似度为 0，否则为 1。另一种则依据具体情况而定，不是简单的非此即彼划分，而是针对不同的属性值间不同的关系给以具体的定义。前者其实是质上的，即非此即彼的二值分割；而后者则是量上的，进一步细化值间的区别。一般来讲，前者定义通用，适于种种情况；而后者则是由人来预定义，与领域知识相关的，从而专用性强。两方法各有自己的适用范围。

3. 有序属性的相似度

有序属性介于数值和枚举型属性之间，也介于定性和定量之间。属性值有序，可以赋予不同等级值间有不同的相似度。和枚举型属性相比，有序属性规整性强。假设属性值分为 n 个等级，则等级 i 和等级 j ($1 \leq i, j \leq n$) 之间的相似度可定义为 $1 - \frac{|i - j|}{n}$ 。

数值属性、有序属性和枚举型属性之间可以相互转化，有时一个属性可以由数值属性来刻划，也可以由有序属性来刻划，比如学生成绩可以由 0 到 100 的分数来反映也可以用 A、B、C 来反映，只不过刻划的方式不同。

我们要计算范例之间的相似度，必须考虑组成一个范例的各个属性相似度综合在一起形成的效应。范例的相似度也常常是通过距离来定义的。常用的典型距离定义有：

1. 绝对值距离(Manhattan):

$$d_{ij} = \sum_{k=1}^N |V_{ik} - V_{jk}|$$

(5.2)

其中 V_{ik} 和 V_{jk} 分别表示范例 i 和范例 j 的第 k 个属性值。

2. 欧氏距离(Euclidean)

$$d_{ij} = \sqrt{\sum_{k=1}^N (V_{ik} - V_{jk})^2}$$

(5.3)

3.麦考斯基距离

$$d_{ij} = \left[\sum_{k=1}^N |V_{ik} - V_{jk}|^q \right]^{1/q}, q > 0$$

(5.4)

上面的距离定义还只是属于平凡的定义，把各属性所起的作用相同。事实上各属性对一个范例整体上的相似度有不同的贡献，因而还需加上权值。即上式可以写成：

$$d_{ij} = \sum_{k=1}^N w_k d(V_{ik}, V_{jk})$$

(5.5)

其中， w_k 为第 k 个属性权值大小，一般要求 $\sum_{k=1}^N w_k = 1$ 。 $d(V_{ik}, V_{jk})$ 表示第 i 个范例和第 j 个

范例在第 k 个属性上的距离，它可以为前面定义的典型距离，也可以是其它的距离定义。有了距离的定义，就可以类似地得到两个范例间相似度定义：

$$SIM_{ij} = 1 - d_{ik} \quad (\text{当 } d_{ij} \in [0, 1])$$

$$\text{或为: } SIM_{ij} = \frac{1}{1 + d_{ij}} \quad (\text{当 } d_{ij} \in [0, \infty) \text{ 时})$$

(5.6)

此外，Cognitive System 公司开发的 ReMind 软件所使用的下式计算相似度：

$$\sum_{i=1}^n w_i \times \text{sim}(f_i^l, f_i^r) / \sum_{i=1}^n w_i \quad (5.7)$$

其中， w 是反应特征重要性的权值， sim 是相似度函数， f_i^l , f_i^r 分别是源范例和范例库中范例的第 i 个特征的值。

5.8 范例的复用

把检索到的旧范例的解答复用到新问题或新范例之中。通过所给问题和范例库中范例比较得到新旧范例之间的不同之处，然后回答哪些解答部分可以复用到新问题之中。对于简单的分类问题，仅需要把旧范例的分类结果直接用于新范例，它无需考虑新旧范例之间的差别。而对于问题求解类的问题，则需要对领域知识的深入理解，根据范例之间的不同对问题进行调整，可以是对整个解的某项作一些调整，也可以对整个解的进行微调。

从复用的信息内容来看，主要有两种类型：结果的复用和方法的复用。对与结果的复用，当旧范例的解答结果需要调整时，它依据一些转换操作知识，把旧范例中的种种可能解转换为新范例中相应的解。方法的复用则关心旧范例中问题的求解方法，而不是其解答的结果。用那一种方法依具体问题而定。

当复用阶段产生的求解结果不好时，需要对其进行修正。修正的第一步是对复用结果进

行评估，如果成功，则不必修正，否则需要对错误和不足进行修正。进行结果评估，可以依据它在实际环境中运行后的反馈，也可以通过向专家询问完成。等待反馈有时可能需要花一段时间，如等待病人治疗的效果如何。但如是工程中的在线应用，则可以马上返回结果。

过去的情景不可能与新情景完全一样，因此对于问题求解型的 CBR 系统必须修正过去的问题解答以适应新的情景。修正过程的输入是当前的问题描述和不太正确的建议解，输出是更适合当前情景的较好的解答。

简单的修正只需对过去解中的某些组成部分进行简单的替换，复杂地修正甚至需要修改过去解的整体结构。修正可以在新解的形成过程中完成，也可能是当新解在执行过程中出现了问题再来做。修正一般有这样几种形式：在旧解中增加新的内容，或从旧解中删去某些内容，或对旧解中的某些内容进行替换，或对旧解中的某些部分进行重新变换。

修正有四类方法：替换法（substitution）、转换法（transformation）、特定目标驱动法（special-purpose adaptation and repair），以及派生重演法（derivational replay）。

1. 替换法

替换方法把旧解中的相关值作相应替换而形成新解。此类方法包括如下 6 种：

(1)重新例化（reinstantiation）：这是一种很简单的替换操作，仅仅是用新的个体替换旧解中的个体。例如，川菜设计系统 CHEF，在根据牛排炒甘蓝菜来设计一道鸡肉炒雪豆菜，它就是把该菜谱中的所有牛排替换成鸡肉，把甘蓝替换成雪豆。

(2)参数调整（parameter adjustment）：这是一种处理数值参数的启发式方法。它和具体的输出与输入参数间的关系模型（输入发生什么变化，会导致输出产生怎样的相应变化）有关。

(3)局部搜索（local search）：使用辅助的知识结构来获得替换值。例如，设计点心时缺少桔子，则可使用此法在一个水果语义网知识结构中搜索一个与桔子相近的水果如苹果来代替。

(4)查询（query）：用带条件的查询在范例库或辅助知识结构中获取要替换的内容。

(5)特定搜索（specialized search）：同时在范例库和辅助知识结构中进行查询，但在范例库中查询时使用辅助知识来启发式指导如何搜索。

(6)基于范例的替换（case-based substitution）：使用其它的范例来建议一个替换。

2. 转换法

转换法包括：常识转换法（common-sense transformation）：使用明白易懂的常识性启发式从旧解中替换、删除或增加某些组成部分。典型的常理转换法是，“删去次要组成部分”。模型制导修补法（model-guided repair）：通过因果模型来指导如何转换。故障诊断中就经常使用这种方法。

3. 特定目标驱动法

这种方法主要用于完成领域相关以及要做结构修改的修正。该法使用的各种启发式需要根据它们可用的情景进行索引。特定目标驱动的修正启发式知识一般通过评价近似解作用，并通过使用基于规则的产生式系统来控制。

4. 派生重演

上述方法所做的修正是在旧解的解答上完成的。重演方法则是使用过去的推导出旧解的方法来推导出新解。这种方法关心的是解是如何求出来的。同前面的基于范例替换相比，派生重演使用的则是一种基于范例的修正手段。

5.9 范例的保存

范例插入范例库的过程类似于检索过程。“remember”有两种含义：“记住”和“回忆”。回忆即检索，记住即存储或插入。插入要调用索引选择过程，以决定范例被索引的方式。插

入算法使用这些索引来把范例插入范例库中适当的地方。一般来说,插入工作所做的搜索和检索相同。插入算法搜索的目的是找到一个可插入范例的地方,而检索的目的是为了找到相似的范例。当检索算法找到了相似的范例后就进行范例排位,而插入算法则是插入源范例并根据需要重新组织范例库结构。

新问题得到了解决,则形成了一个可能用于将来情形与之相似的问题。这时有必要把它加入到范例库中。这是学习也是知识获取。此过程涉及选取那些信息需要保留,以及如何把新范例有机的集成到范例库中,并且会涉及范例库的组织和管理方面的知识。

在决定选取范例的哪些信息进行保留时,一般要考虑以下几点:和问题有关的特征描述;问题的求解结果;以及解答为什么成功的原因及解释。

把新范例加入到范例库中时,需要对它建立有效的索引,这样以后才能对它作出有效的回忆。索引应做到:与该范例有关时能快速回忆到,与它无关时不应回忆出。为此,可能要对范例库的索引内容甚至结构进行调整,如改变索引的强度或特征权值。

随着时间的推移,范例库会越来越大,这将浪费存储空间,增加检索的时间。因此必须对范例库进行有效的组织和管理。

在上述检索(Retrieval)、重用(Reuse)、修正(Revise)和保存(Retain)四个过程是基于范例推理的关键步骤。由于它们的英文都是以R开始的,因此,CBR的推理过程也称为4-R过程。

5.10 基于例示的学习

基于例示的学习(instance-based learning: IBL),是一种与基于范例的学习紧密相关的归纳学习方法[Aha 1991]。基于例示的学习算法的思想是,存储有过去的已分类的例示,当对新来的输入进行分类时,算法在已分类例示中寻找与输入情况最相似的例示,然后把该事例的类别作为对新例示的分类结果。IBL没有用到复杂的索引,仅仅使用特征-值表示方法,也不做范例修正操作,但它却是一种非常有用的方法。

在实例学习(learning from examples)或有监督学习(supervised learning)研究中,人们提出了各种学习概念的表示方法,如规则、决策树、神经网络等等。这些方法的共同特点是利用对训练实例的抽象泛化来预测未来的新事例。最近邻(nearest neighbor)方法,也称作基于例示学习,它不对训练事例抽象泛化,而是直接以典型事例来表示概念。它预测新事例的方法是,根据相似性原理(即假设相似的事例有相似的分类结果),在已存储的例示集中寻找一个或若干个最相似的例示,然后综合这些例示的已有分类结果以形成预测分类。最近邻方法的学习方式是渐增式的;在连续值属性的预测分类中,和其它学习方法相比较,一般情况下它具有最好的预测准确度[Biberman 1994]。

最近邻方法的一般形式是k-近邻(简称k-NN),k表示若干个近邻。K-NN的应用中存在几个问题:怎样从例示集或例示库中选择若干个与要预测的新事例相似的例示,如何综合评价这些相似例示的结果以形成当前事例的预测值。前者又包括如何定义两个例示间的相似度,如何确定K值大小。Weiss等对符号或离散值属性给出了较好的解决方法[Weiss 1991]。Aha给出了一种交叉验证(cross validation)的k值确定方法[Aha 1997]。

5.10.1 基于例示学习的任务

基于例示学习中,例示表示成属性值对,每一个例示具有几个属性,属性缺值是允许的。

属性集对应一个多维空间。这些属性中有一个是类别属性，其余属性是条件属性。基于例示学习算法学习多个、重叠的概念描述。但一般情况下只涉及一个类别属性，并且类别是不相交的，输出是最基本的。

基于例示学习中概念[描述]是从例示到类别的函数：给定例示空间中的一个例示，它反映一个预测这个例示类别标记的分类。基于例示的概念描述，包括存储的例示集合，可能还包括它们在过去分类过程中的性能，例示集，可以在每个实例处理后发生变化。

分类函数：输入相似性函数和概念描述中例示的分类性能记录。输出类别标记 I。概念描述更新器：它维护分类性能记录，决定哪些例示应加入概念描述中。基于例示学习认为相似的例示有相似的类别，因此，存在将新实例按其大多数相似的邻居的类别进行分类。同时，基于例示学习假定在缺少先验知识的条件下，所有的属性对分类决策的贡献是相同的，即相似函数中权重相同。这一偏置要求对每个属性在其值域内归一化。

基于例示学习与大多数有导师学习算法不同，它不构造决策树和决策归纳之类的明确的精炼的模式。后者通过泛化表示实例。分类时采用简单的匹配，而基于例示学习在实例表示所做的工作很少，几乎不进行泛化，而对后继例示的分类需要的计算较多。

基于例示学习的性能可以从以下几个方面考虑：

- 泛化能力：指表示的可描述与算法的可学习性，IBL 算法是 PAC 学习的，其概念边界由若干有限个有限大的闭超曲线组成。
- 分类精度。
- 学习速度
- 协作代价：指用一个训练实例更新概念描述的开销，包括分类的开销。
- 存储要求：概念描述的大小，IBL 中指用于分类决策所需的实例数。

5.10.2 IB1 算法

IB1 算法的思想相当简单，它使用最近邻例示的类别标记作为预测值。必须指出，如果给定的属性在逻辑上不足以描述目标概念，IB1 将不会成功。

算法 5.1 IB1 算法

1. $CD \leftarrow \phi$ //CD=概念描述
2. For each $x \in \text{Training Set}$ do
3. for each $y \in CD$ do
4. $\text{sim}[y] \leftarrow \text{similarity}(x, y)$.
5. $y_{\max} \leftarrow \text{some } y \in CD \text{ with maximal sim}[y]$
6. if $\text{class}(x) = \text{class}(y_{\max})$
7. then $\text{classification} \leftarrow \text{correct}$
8. else $\text{classification} \leftarrow \text{incorrect}$
9. $CD \leftarrow CD \cup \{x\}$

在一般统计假定条件下，近邻决策策略在最坏的情况下错分率至多是最佳贝叶斯 (optimal Bayes) 的两倍。这个结果是建立在样本数不受限制的条件作出的，因而较弱。下面我们来对错误估计进行分析。

定义 5.6 R^n 中点 x 的 ϵ -球是到 x 的距离小于 ϵ 的点集。 $\{y \in R^n \mid \text{distance}(x, y) < \epsilon\}$ 在二维空间中 ϵ -球近似为 ϵ^2 。

定义 5.7 设 X 是 R^n 中固定概率分布的点集。 X 的子集 S 是 X 的 $\langle \epsilon, r \rangle$ 网，如果对 X 中的

所有 x ，除了概率不超过 r 的部分外， S 中存在 s 使 $|s-x| < \varepsilon$ 。

引理 5.1 设 ε, γ 是确定的，小于 1 的正数，在任何确定的概率分布下， $[0-1, 0-1]$ 上包含 N 个实例的随机样本， $N \geq \lceil \sqrt{2} / \varepsilon \rceil^2 / \gamma \times \ln \lceil \sqrt{2} / \varepsilon \rceil^2 / \delta$ ，将形成一个可信度大于 $1-\delta$ 的 $\langle \varepsilon, \gamma \rangle$ -网。

证明 将单位面积分成大小相等的 k^2 小方块，每一个对角线长度不大于 ε ，则有 $k = \lceil \sqrt{2} / \varepsilon \rceil$ ，并且小方块中所有实例距离在 ε 之内。设 S_1 是概率不小于 γ / k^2 的小平面的集合。设 S_2 是剩下的小平面。任意实例 i 不在 S_1 中的概率至多 $1 - \gamma / k^2$ 。样本中 N 个实例均不在 S_1 中的概率至多 $(1 - \gamma / k^2)^N$ 。 S_1 中任意平面不包含 N 个实例的样本的概率至多为 $k^2 (1 - \gamma / k^2)^N$ 。

由 $(1 - \gamma / k^2)^N < e^{-Nr/k^2}$ ，则 $k^2 \times (1 - \gamma / k^2)^N < k^2 \times e^{-Nr/k^2}$ 。我们确保这样的概率少于 δ 。

此时有 $N \geq (\lceil \sqrt{2} / \varepsilon \rceil^2 / \gamma) \times \ln \lceil \sqrt{2} / \varepsilon \rceil^2 / \delta$ 。因此， S_1 中的平面包含 S 的样本实例。在 S_2 中的所有小方块的概率小于 $(\gamma / k^2) \times k^2 = \gamma$ 。

这一证明可以推广到 R^N 维上，此时 R^N 上需要的样本数为 $(\lceil \sqrt{N} / \varepsilon \rceil^N / \gamma \times \ln \lceil \sqrt{N} / \varepsilon \rceil^N / \delta)$ ，它随维数的指数次方增长。□

定义 5.8 对任何 $\varepsilon > 0$ ，集合 C 的 ε -内核(core)是包含在 C 中的、并且属于 C 中所有点的 ε -球的点集。

定义 5.9 C 的 ε -近邻是到 C 中某些点的距离在 ε 之内的点集。

定义 5.10 集合 C' 是 C 的 $\langle \varepsilon, \gamma \rangle$ -近似，如果忽略一些概率小于 γ 的集合， C' 包含 C 的 ε -内核，并且被 C 的 ε -近邻包含。

IBL 算法中，每一个实例都被保存，几乎总是覆盖所有的目标概念的近似正确定义。“几乎总是”意味着概率大于 $1-\delta$ ， δ 是任意小的正数。“近似正确”确保生成的概念是目标概念的 $\langle \varepsilon, \gamma \rangle$ -近似， ε, γ 同样是任意小的正数。

引理 5.1 证明 IBL 生成的概念与目标概念的近似程度。特别的除了概率小于 γ 的情形外，IBL 几乎总是覆盖概念的 ε -内核与 ε -近邻之间的集合。

定理 5.1 设 C 是 $[0-1, 0-1]$ 上封闭曲线所围绕的区域，对给定的 $1 > \varepsilon, \gamma, \delta > 0$ ，IBL 算法覆盖 C' ：

$$(\varepsilon\text{-core}(C) - G) \subseteq (C' - G) \subseteq \varepsilon\text{-近邻}(C) - G。$$

其中 G 是概率小于 γ 的集合。并且这一覆盖的可信度是 $1-\delta$ 。

证明 由上一引理， $N \geq (\lceil \sqrt{2} / \varepsilon \rceil^2 / \gamma) \times \ln \lceil \sqrt{2} / \varepsilon \rceil^2 / \delta$ 时，任意 N 个随机选择的样本将形成 C 的 $\langle \varepsilon, \gamma \rangle$ -网（可信度 $1-\delta$ ）。

设 C' 是 IBL 生成的预测属于 C 的点集。设 G 是 $[0-1, 0-1]$ 上 N 个样本中任何一个距离不在 ε 之内的点集。

除 G 外， C 的 ε -内核包含在 C' 中，（也在 $C'-G$ 中），设 p 是 C 的 ε -内核中、但不在 G 的任意点。 s 是 p 的近邻，由 s, p 的距离小于 ε ， p 在 ε -内核中， s 在 C 中， s 被正确地预测是 C 的成员。同时， p 也是 C' 的成员，所以 $(\varepsilon\text{-内核}(C) - G) \subseteq (C' - G)$ 。

同时，如果 p 在 C 的 ε -近邻外，故 p 在 $C'-G$ 之外。设 s 是 p 的近邻，如果 p 不在 G 中，则 s 不在 C 中。由 s 可正确预测 p 不是 C 的成员。由于没有点在 C 的 ε -近邻之外，被 C' 预测成 C 的成员，（除 G 中的点之外），故 $(C' - G) \subseteq (\varepsilon\text{-近邻}(C) - G)$ 。

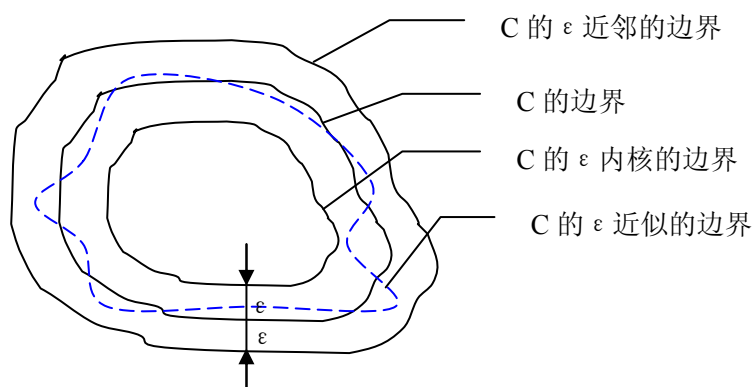


图 5.5 近邻的边界

该定理表明，对给定的可信度，如果训练例子足够，将形成 C 的 ϵ -内核与 ϵ -近邻之间的 C 的近似（忽略概率小于 γ 的情形）。 C' 不能正确分类 p 的情形是： $p \in C$ 的 ϵ -近邻，但 $p \notin C$ 的 ϵ -内核； $p \in G$ 。

定义 5.11 对概率分布 P 的类，概念类 C 是多项式可学习的，当且仅当存在算法 A ，多项式 γ 使任意 $0 < \epsilon$ ， $\delta < 1$ ，和任意 $C \in \mathcal{C}$ ，如果按某确定概率分布 $p \in P$ ，超过 $\gamma(1/\epsilon, 1/\delta)$ 个样本被选定，那么，在可信度至少 $1 - \delta$ 的前提下， A 将输出 C 的假设，这一假设同样是 C 的成员，并且与 C 的差别概率小于 ϵ 。

定理 5.2 设 C 是 $[0-1, 0-1]$ 上所有长度小于 L 的曲线所封闭的区域对应的概念类， P 是可以由上述 B 所封闭的概率密度函数表示的概率分布类 \mathcal{E} 。则 C 是在 P 下用 IB1 多项式可学习的。

证明 由上引理，如果 C 的边界长度小于 L ，则 C 的 ϵ -内核和 ϵ -近邻之间的面积小于 $2\epsilon L$ 。 $\alpha = 2\epsilon LB$ 是该区域[在概率下]上边界。 C' 导致的错误小于 $\alpha + \gamma$ 。如设 $\gamma = \alpha = \epsilon/2$ ， $\epsilon = \epsilon/4LB$ 。替换不等式中的 γ ， ϵ ，即可得证。

由该定理，IBL 需要的例示数是关于 L 和 B 的多项式的。通过以上分析，我们可以得出：

- 如 ϵ -内核为空， C' 可以是 C 的 ϵ -近邻的任意子集， C 的形状很小， ϵ 足够大，此时，IBL 对 C 的近似程度不好。
- 目标概念的边界程度增加时，学习需要的实例的期望数增加。
- 除了大小不到 γ 的情形，错误的正例包括在 C 和 C 的 ϵ -近邻之间的外圈，错误的反例包含在内圈。
- IBL 不能区分任何包含在 ϵ -内核， ϵ -近邻之间的概念，因此，这一方法不能反映概念边界小的变动。
- 所有的结论可以扩展出任意维空间。

5.10.3 降低存储要求

IB1 算法中，只有 C 的 ϵ -近邻与 ϵ -内核之间的例示是产生概念边界的精确性近似。其余的例示并不能区分边界的位置。因此，只保存那些有用的例示将节省大量的存储空间。不过，在缺乏关于概念边界的全部知识下，这样的实例集合是没法知道的，但可以通过 IB1 算法中被错误分类的实例近似表示，这就是 IB2 的算法的思路。

算法 5.2 IB2 算法。

1. $CD \leftarrow \phi$ //CD 是概念描述
2. For each x Training Set do
3. for each $y \in CD$ do
4. $\text{sim}[y] \leftarrow \text{similarity}(x, y)$
5. $y_{\max} \leftarrow \text{some } y \in CD \text{ with maximal sim}[y]$.
6. if $\text{class}(x) = \text{class}(y)$
7. then **classification** \leftarrow **correct**
8. else
8. **classification** \leftarrow **incorrect**
9. $CD \leftarrow CD \cup \{x\}$.

与 IB1 不同的是, IB2 中只保存那些被错分的实例, 而大多数被错分的实例出现在 ϵ -近邻与 ϵ -内核之间 (ϵ 合理地小), 与概念边界非常接近。实例到概念边界的距离变化较大时, IB2 中存储空间减少非常明显。

噪声增大时, IB2 的精度与 IB1 的下降更快, 这是因为这些带噪声的实例总是被错分, IB2 中保存的实例只有小部分是设有噪声的, 那些带噪声的例子用于分类决策自然效果不好。

IB3 是对 IB2 的改进, 使其对噪声不敏感, 思路是用“选择过滤器”确定哪些实例应该保存起来用于将来的分类决策。

算法 5.3 IB3 算法。

1. $CD \leftarrow \phi$
2. For each x Training Set do
3. for each $y \in CD$ do
4. $\text{sim}[y] \leftarrow \text{similarity}(x, y)$
5. if $\exists \{y \in CD \mid \text{acceptable}(y)\}$ then
6. $y_{\max} \leftarrow \text{some acceptable } y \in CD \text{ with maximal sim}[y]$
7. else
8. $i \leftarrow$ a randomly-selected value in $[1, |CD|]$
9. $y_{\max} \leftarrow$ some $y \in CD$ that is the i -th most similar instance to x
10. if $\text{class}(x) \neq \text{class}(y_{\max})$
11. then
12. **classification** \leftarrow **correct**
13. else
14. **classification** \leftarrow **incorrect**
15. $CD \leftarrow CD \cup \{x\}$
16. for each y in CD do
16. if $\text{sim}[y] \geq \text{sim}[y_{\max}]$
17. then
18. update y 's classification record
19. if y 's record is significantly poor
20. then $CD \leftarrow CD - \{y\}$.

在 IB3 中, 对每一个保存的例示维护一个分类记录 (如分类测试中正确分类数和错误分类数)。分类记录概括了每一个例示对当前训练实例的分类性能, 同时表征将来的性能。同时, IB3 进行测试确定哪些例示是好的分类器, 哪些相当于噪声, 后者将从概念描述中删除。对每

一个训练实例 i ，分类记录将更新所有至少与 i 最相似的“可接受”的近邻。

如果保存的例示没有一个是可接受的，我们采用假定至少一个例示是可接受的策略，产生一个 $[1, n]$ (n 是保存的例示数) 之间的随机数 γ ， γ 个最相似的例示的分类记录将被更新。如果至少一个例示是可接受的，将更新 i 为中心， I 与它的最近的可接受邻居之间的距离为半径的超球内的被保存例示的分类记录。

IB3 使用信任区间来描述，来确定一个实例是可接受的，中等的，或者噪声。信任区间由实例当前分类精度，其类别固有频率确定。如果精度的下限高于自身被正确分类的概率下限，则是可接受的；如果精度的上限小于自身被正确分类的概率的下限，则是应去掉的噪声。如果两个区间重叠，有待于进一步训练确定。

IB3 学习性能对例示的不相干特征数相当敏感。同时，需要的存储空间和学习速度随维数的指数次增加。IB3 不能表示重叠的概念。假定所有的类只属于一个类，当然，每次我们可以只学习每个概念的单独描述。

基于例示学习具有如下优点：简单；鲁棒性相对较好；概念偏置相对宽松，能渐增地学习概念的曲线线性近似，在偏置不能满足目标概念时，基于例示的方法比其它学习算法速度更快。尤其是目标概念的边界不与属性维平行时；基于例示学习算法的更新代价较低。基于例示学习更新的代价包括分类。它要求有 $(|A|*|N|)$ 个属性被检查，其中 $|N|$ 是被保存的实例数， $|A|$ 是特征数，而 C4.5 中对一个新的实例计算复杂度为 $O(|I|*|A|^2)$ 。同时，并行处理和索引技术可进一步降低基于例示学习的计算复杂性。

5.11 范例工程

近十年在这方面的理论和应用表明，范例的途径总是和特定领域相关的。必须注意这两个问题：

- 修正范例在范例库中的组织，使其能够有效地和高效地在将来的推理中重用。
- 范例工程 (case engineering) 自动化：根据已有的信息自动抽取范例。

范例工程是指设计合理的范例库，生成与应用领域相关的知识的部件，包括范例的结构，范例的组织，范例的检索（如索引机制，相似性度量），范例使用的规则，范例的修正与保存。

怎样选择和抽取基本的范例初始化范例库也是非常重要的。通常，抽象范例必须手工的或自动的从具体范例生成。手工的生成无疑将陷入专家系统知识的瓶颈。自动的生成这些抽象范例是一个从具体范例映射的过程，实质是数据约简。

在惰性学习 (Lazy Learning) 中，一般数据是推迟处理的，没有明显的学习阶段，不会得到关于数据的模型。而 KDD 处理的是海量数据，存储和检索它们的代价是很大的。一般认为，惰性学习的方式是不适合处理大数据量的。我们的思路是将惰性学习与积极学习结合起来：从原始的海量数据中得出粒度适中的若干小模型（这些模型的个数比原始数据要小的多），每个小模型是关于部分原始数据的，所有的小模型反映了整个数据。在测试的阶段，CBR、近邻算法等惰性学习的算法不使用原始的数据，而直接访问这些小模型，如果测试数据与其中的某个模型相匹配，则返回这个模型的某些特征作为结果。这种惰性学习与积极学习相结合的方法，必须具备如下的特点：

- 学习算法应该是渐增的：增加新数据时，原来的模型能够通过修正重用，添加数据同时可以构造模型，这样将减少训练时间；

- 模型是层次性的：粒度越小的模型概括的数据越少，精度越高；粒度越大的模型概括的数据越多，精度越低；粒度越小的模型与粒度越大的模型之间是层次关系的；上一层的模型可以作为下一层的索引；
- 不同层次的模型的格式未必是相同的，这一方面是领域知识所要求的，另一方面用户和算法对不同层次知识的要求也不尽相同；
- 模型的表示要便于计算机使用，也要便于用户理解。
- 用户可以根据具体的问题选用不同粒度的模型。

一般认为经典的近邻算法的复杂度比较高，因为要对新实例预测类别，必须访问所有的实例。将所有的实例存放在内存中的开销是很大的。关键是怎样减少需要存储的范例数。通常的做法是将所有的实例都作为范例，实际上从实例集中选择恰当的部分作为范例还可以提高预测精度。

在模式识别和基于例示的学习中，减少近邻算法中的例示数问题有时候称为引用选择问题（reference selection problem）[Dasarathy 1991]，其目标是为了提高预测的精度选择恰当地训练实例的数量和质量。已有的工作包括存储错误的例示，如 Condensed Nearest Neighbor algorithm, Reduced Nearest Neighbor algorithm, IB2[Aha 1990]；存储典型的例示[Zhang 1992]；仅存储那些被其它训练例示得出的分类器正确分类的例示[Wilson 1972]；由领域知识确定的方法；存储平均例示或抽象例示[Aha 1990]；利用遗传算法选择例示[Kurtzberg 1987]，综合的方法[Voisin 1987]。在 Aha 的研究表明，利用有限个特征加权的例示可以提高分类的精度，IB3、IB4 算法的效果比常规的近邻算法 IB1 要好。

选择部分例示获得更好的分类性能是可能的，因为如果忽略那些带有噪音的例示，就可避免训练过度的问题。此外，较少的例示意味简单的偏置假设，优先简单的模型符合机器学习理论的原则。

研究基于范例的推理的动机之一是它能减轻开发智能系统的复杂度和难度。进一步，为了减轻基于范例推理系统的开发工作，设计基于范例推理的开发工具，将使专家能直接参预范例的获取等知识工程问题。这类似于专家系统开发工具的作用。基于范例的推理工具可分为两种：通用型和特定任务型。通用型基于范例的推理工具可支持范例的管理、检索、评估，以及系统结构上的修正等。FABEL 就是这样一个系统。特定任务型 CBR 工具 目前有这样几类：①基于范例的辅助决策系统设计工具，如 Design-MUSE[Domeshek 1994]；②基于范例的咨询系统设计工具，如 REPRO[Mark 1996]；③基于范例的教学系统设计工具，如 GuSS[Burke 1996]。

另外除了上述项目，市场上销售的 CBR 工具或外壳也有不少。这些 CBR 外壳各有特点，如提供了近邻检索方法或自动生成决策树的范例检索机制；允许用户在检索时交互式地向系统提供所需信息；提供了能方便用户创建和编辑范例库的友好界面，同时也具有从数据库中提取所需信息的功能。这类 CBR 外壳有 CBR Express, Esteem, Kate, ReMind, CBR Works, ReCall, Eclipse, ART* Enterprise 等。通过使用 CBR 外壳，系统构造者就只需负责收集范例的工作了。

5.12 范例约简算法

叶施仁、史忠植提出了一种基于格（lattice）的高效约简算法 $I_{\text{NREDUCTCLS}}$ ，旨在将数据形成粒度较大的超范例。 $I_{\text{NREDUCTCLS}}$ 生成最小 E-集和最大 E-集之间的超元组，从超元组中能够得到与原始数据相同的、甚至更好的决策[叶施仁 2001]。超元组 h 表示成三元组 $(|h|, \{x_{\text{dsp}}\}, \{\text{child}_i\})$ ，其中 $|h|$ 表示 h 中包括的简单元组数目， $\{x_{\text{dsp}}\}$ 是每个属性的表示， $\{\text{child}_i\}$ 是指向 h 包含的元组的指针的集合。超元组是特征空间中的超立方体，通过分析它们边界的密度朝样本密集区域移动，因而具有很好的代表性和概括能力。算法是渐增的，次优的。在该方法中，利用各个维上的边界处密度控制超元组对应的超立方体在特征空间中调整，达到次优的覆盖能力和代表性，前者是约简率的保证，后者是正确性的保证，并且通过引入格的方法可以从理论上分析该方法的有效性和合理性。

定义 5.12（超）元组 h 通过属性区间表示，具体地说，对于归一化后的有序属性 x 表示成：

$$\text{mid}(h(x)) = \sum_{x \in \text{cov}(h)} h(x) / |\text{cov}(h(x))|$$

平均值，记为 x_{avg} ；

上边界 $\max(h(x))$ 简记为 x_{max} ；

下边界 $\min(h(x))$ 简记为 x_{min} ；

上边界的元组数是指在属性 x 的取值在 $[x_{\text{max}} - \Delta, x_{\text{max}}]$ 的简单元组数

$|\text{cov}(\max(h(x)) - \Delta, \max(h(x)))|$ ，记为 $|x_{\text{max}}|$ ，于是超元组 h 的上边界在属性 x 方向上投影

的密度为 $\text{den}_{\text{up}}(h(x)) = |\text{cov}(\max(h(x)) - \Delta, \max(h(x)))| / \Delta$ ，即 $|x_{\text{max}}| / \Delta$ ；类似地

下边界的元组数是指在属性 x 的取值在 $[x_{\text{min}}, x_{\text{min}} + \Delta]$ 的简单元组数

$|\text{cov}(\min(h(x)), \min(h(x)) + \Delta)|$ ，记为 $|x_{\text{min}}|$ ，于是超元组 h 的下边界在属性 x 方向上投影的

密度为 $\text{den}_{\text{down}}(h(x)) = |\text{cov}(\min(h(x)), \min(h(x)) + \Delta)| / \Delta$ ，即 $|x_{\text{min}}| / \Delta$ ；于是，超元组 h 中

属性 x 可以表示成 $x_{\text{dsp}} = (x_{\text{avg}}, x_{\text{max}}, x_{\text{min}}, |x_{\text{max}}|, |x_{\text{min}}|)$

对于离散属性，我们给出超元组 h 中属性 x 所有可能取值的二元组的列表：

$$\{(x_i, |x_i|) | x \in V_x, i = 1, \dots, |V_x|\},$$

其中 $|x_i|$ 是属性取值 x_i 的出现次数。定义 x 中 x_{min} 为赋值次数最少的属性值，即 $|x_{\text{min}}| \stackrel{\text{def}}{=} \min(|x_i|)$ ；如果这样的赋值多于一个，则任取一个。类似地， x_{max} 为赋值次数次少的属性值。

x_{avg} 为赋值次数最多的属性值， $|x_{\text{avg}}| \stackrel{\text{def}}{=} \max(|x_i|)$ 。于是，离散属性 x 可以表示成 $x_{\text{dsp}} = ((x_i, |x_i|)), x_{\text{avg}}, x_{\text{max}}, x_{\text{min}}, |x_{\text{max}}|, |x_{\text{min}}|)$ 。

于是，超元组 h 表示成三元组 $(|h|, \{x_{\text{dsp}}\}, \{\text{child}_i\})$ ，其中 $|h|$ 表示 h 中包括的简单元组数目， $\{x_{\text{dsp}}\}$ 是每个属性的上述表示， $\{\text{child}_i\}$ 是指向 h 包含的元组的指针的集合。特别地，如果某个属性的分布是近似于随机的，即几乎任意地充满该属性的可能的赋值空间，则可以忽略该属性。为了表达方便，称 $|h|$ ， $\{x_{\text{dsp}}\}$ 和 $\{\text{child}_i\}$ 为 h 的描述因子。

定义 5.13 超元组 h 关于属性 x 的上边界的相对密度为 $|x_{\text{max}}| / |h|$ 。超元组 h 关于属性 x 的

上下界的相对密度为 $|x_{\min}|/|h|$ 。

设超元组 $h'=(|h'|, \{x'_{\text{dsp}}\}, \{\text{child}'_i\})$, $h''=(|h''|, \{x''_{\text{dsp}}\}, \{\text{child}''_i\})$, 则 h' 与 h'' 合并生成的超元组 h 。设 h' 和 h'' 中相应的描述指标都分别地加上 “'” 和 “''” 符号 (如 h'' 中属性 x 的平均值表示成 x''_{avg})。对有序属性,

$$x'_{\text{avg}} + x''_{\text{avg}} \stackrel{\text{def}}{=} (x'_{\text{avg}} * |h'| + x''_{\text{avg}} * |h''|) / (|h'| + |h''|)$$

$$x'_{\text{max}} + x''_{\text{max}} \stackrel{\text{def}}{=} \max\{x'_{\text{avg}}, x''_{\text{avg}}\}$$

$$x'_{\text{min}} + x''_{\text{min}} \stackrel{\text{def}}{=} \min\{x'_{\text{avg}}, x''_{\text{avg}}\}$$

$$|x'_{\text{max}} + x''_{\text{max}}| \stackrel{\text{def}}{=} \begin{cases} |x'_{\text{max}}| & \text{当 } x'_{\text{max}} + \Delta' > x''_{\text{max}} \\ |x''_{\text{max}}| & \text{当 } x''_{\text{max}} + \Delta'' > x'_{\text{max}} \\ |x'_{\text{max}}| + |x''_{\text{max}}| - \varepsilon & \text{否则} \end{cases}, \text{ 其中}$$

$$\varepsilon = |x'_{\text{max}} \cap x''_{\text{max}}| \cong \left(\frac{|x'_{\text{max}}| + |x''_{\text{max}}|}{2} \right) * \frac{(\Delta' + \Delta'') - \text{abs}(x'_{\text{max}} - x''_{\text{max}})}{\Delta' + \Delta''}$$

(此时上边界重叠, 上式中上边界大小为 $\text{abs}(x'_{\text{max}} - x''_{\text{max}})$, 也可以定义新的 Δ , 根据 h 中包含的元组重新统计)

$$|x'_{\text{min}} + x''_{\text{min}}| \stackrel{\text{def}}{=} \begin{cases} |x'_{\text{min}}| & \text{当 } x'_{\text{min}} + \Delta' < x''_{\text{min}} \\ |x''_{\text{min}}| & \text{当 } x''_{\text{min}} + \Delta'' < x'_{\text{min}} \\ |x'_{\text{min}}| + |x''_{\text{min}}| - \varepsilon & \text{否则} \end{cases}, \text{ 其中}$$

$$\varepsilon = |x'_{\text{min}} \cap x''_{\text{min}}| \cong \left(\frac{|x'_{\text{min}}| + |x''_{\text{min}}|}{2} \right) * \frac{(\Delta' + \Delta'') - \text{abs}(x'_{\text{min}} - x''_{\text{min}})}{\Delta' + \Delta''}$$

(此时下边界重叠, 上式中下边界大小为 $\text{abs}(x'_{\text{min}} - x''_{\text{min}})$, 也可以定义新的 Δ , 根据 h 中包含的元组重新统计)

对枚举型属性,

$$|x'_i + x''_i| \stackrel{\text{def}}{=} |x'_i| + |x''_i|$$

$$|x'_{\text{min}} + x''_{\text{min}}| \stackrel{\text{def}}{=} \min(|x'_i| + |x''_i|)$$

$$|x'_{\text{max}} + x''_{\text{max}}| \stackrel{\text{def}}{=} \max(|x'_i| + |x''_i|)$$

于是, 我们得到

引理 5.2 设超元组 $h'=(|h'|, \{x'_{\text{dsp}}\}, \{\text{child}'_i\})$, $h''=(|h''|, \{x''_{\text{dsp}}\}, \{\text{child}''_i\})$, 则 h' 与 h'' 合并生成的超元组 h

$$h' + h'' = (|h'| + |h''|, \{x'_{\text{dsp}}\} + \{x''_{\text{dsp}}\}, \{\text{child}'_i\} + \{\text{child}''_i\}).$$

当一个简单元组 s 与超元组 h 合并时, 可以把这个简单元组看成一个超元组, 按照上述方法进行合并。事实上, 这里可以分为两种情形:

(1) h 在 x 方向上覆盖了 s , 即 $x_{\min} \leq s(x) \leq x_{\max}$, 此时 h 的边界不要修改, 只需要更新

- 对有序属性, $x_{\text{avg}} = (x_{\text{avg}} * |h| + s(x)) / (|h| + 1)$, 对枚举属性, 如果 $x_i = s(x)$, $|x_i| \leftarrow |x_i| + 1$;
- 如果 s 在属性 x 方向上落在上边界或者下边界, 则更新边界区包括的元组数;

(2) h 在 x 方向上没有覆盖 s 。此时, 如果先调整 h 的边界, 然后再更新 h 的其它描述因子, 将会带来很大的计算量, 这会很大程度上影响算法的性能。事实上, 如果 s 与 h 的距离足够近,

在 x 方向上, s 将位于 x 的上边界 (或下边界) 附近, 由于定义的边界是一个弹性的区域概念, 我们可以近似地认为 s 落在边界内, 推迟调整 h 的边界。这样只需在相应边界内进行简单的累加: $|x_{\max}| \leftarrow |x_{\max}| + 1$, 或者 $|x_{\min}| \leftarrow |x_{\min}| + 1$ 。

算法 5.4 基于格的超范例 (元组) 生成算法渐增式约简算法 $I_{\text{NREDUCTCLS}}$ 。

1. 根据可用内存空间大小, 随机在样本 S 中选择 N 个不互为近邻的点作为初始的超元组的中心 (它们之间的距离不小于预先规定的阈值 ξ), 超元组的标记为相应点的类别标记。每个维上 $x \pm \Delta$ 为上边界和下边界, $H = \{h_1, h_2, \dots, h_N\}$;
2. 对样本依次进行扫描, 对每个点 p 找到与其最近的类别相同超元组 h_i , 设 p 与 h_i 之间的距离为 $\text{dis}(p, h_i)$ 。
 - 2.1 如果 $\text{dis}(p, h_i) \leq \xi$, 或者 p 被 h_i 覆盖, 则
 - i. 将 p 标记为 i
 - ii. h_i 中加入指向 p 的指针 (或 p 的 ID)
 - iii. 更新 h_i 的描述因子
 - 2.2 否则如果内存空间允许, 则
 - i. 将 p 作为一个新的初始的超元组加入 H
 - ii. $N \leftarrow N + 1$
 - iii. 将 p 标记为 N
 - iv. h_i 中加入指向 p 的指针
 - 2.3 否则, 将 p 视为独异点, 将其标记为 0
3. 在步骤 2 的扫描过程中的某些阶段, 或者扫描结束后, 进行如下操作
 - 3.1 根据 h_i 中包含的元组、边界密度, 调整其边界, 重新统计其描述因子;
 - 3.2 合并边界重叠的或者中心距离小于 ξ 的类别相同超元组;
 - 3.3 根据超元组的密度 (或包括的元组数目) 排序, 去掉一些密度很低的超元组, 并将其中被包括对点标记为 0;
 - 3.4 如果类别不同的超元组 h_i 与 h_j 覆盖的区域重叠, 则将密度比较低的超元组去掉重叠的部分, 减小其覆盖范围;
4. 转步骤 2 对样本重新进行扫描和标记, 直到样本中的标记不发生变化, 或者所有超元组的边界不改变, 或者满足用户的某些标准;
5. 对标记为 0 的样本重复步骤 1 到步骤 4, 它得到的密度比较高的超元组加入以前获得的、标记相同的超元组中。

5.13 中心渔场预报专家系统

心理学研究表明, 人类决策喜欢而且善于利用范例作决定, 但人的记忆上的限制又使得人常常难以正确回忆出适当的范例, 特别是在范例数目很多时。和人相比, 计算机则正好在记忆方面具有优势, 能存储大量的范例以及较好地回忆出相关的范例。把人和机器的各自特长结合在一起, 正是基于范例的决策支持 (case-based decision aiding) 的研究目的。

应用现代科学技术开发海洋资源, 养护和管理好海洋渔业资源, 实现海洋渔业可持续发展已经为世界各国广泛重视。计算机技术、遥感技术、自动化技术、地理信息系统技术, 不仅可以为海洋渔业开发和管理提供宏观决策信息, 还能在微观上为海洋渔业资源的利用作出具体指导。

为了更好地对我国海域内渔业资源利用与开发, 强化高新技术对未来海洋专属经济区渔

业资源养护、利用和管理技术的支撑作用，我们在 863 计划支持下开展了海洋渔业遥感信息与资源评估服务系统技术及系统集成、示范试验的研究工作。并以东海（北纬 25 度到 34 度，东经 130 度以西海区）作渔情预报示范区，为外海渔业资源以及远洋渔业资源的利用与开发打下良好的基础。渔情预报利用数据采掘、基于实例的学习、专家系统等技术，在遥感渔业分析技术、海洋渔业服务地理信息系统技术的支撑下，综合领域专家的知识和经验，建立渔情分析与预报的学习和决策支持系统。

5.13.1 问题分析与范例表示

在该系统中，我们的任务除了对渔业资源作出评估（中长期的预报）外，重要的工作是对渔况作出短期的预报（中心渔场）。渔场是指在一定季节海洋鱼类或海产经济动物密集，可用捕捞工具进行生产并具有开发利用价值的海区。中心渔场是指具有较高的平均网产和总渔获量的区域。准确预报中心渔场的位置和大小能够直接地提高渔业生产的产量和效率，具有很大的经济效益。

尽管我们已经初步知道，鱼类的洄游以及中心渔场的形成受到这几个因素的制约：海水温度（包括海洋表面温度，海洋底层温度）；台站数据，如海水盐度，盐度梯度，长江径流量，风向，风速等；海洋叶绿素浓度。但是，鱼类的洄游规律受很多因素制约，变化非常复杂，难以用传统的数学方法和模型描述。同时专家关于中心渔场规律的知识是不精确的，不完全的。值得庆幸的是，我们已经收集了 20 来年东海的渔况海况数据，这是非常宝贵的资料，因此可以从中挖掘出许多有用的信息和知识，根据历年的情况来分析、预测中心渔场的趋势。整个系统采用了基于范例推理（CBR）的方案。因为 CBR 非常适合应用于系统已存在大量历史数据，专家通过实例来描述他们的领域，问题未被完全理解，可用的领域知识很少，系统中有很多例外的规则的情形。

由于大多数海况信息是以周为单位收集的，同时为了便于处理和计算，我们根据实际情况对需求进行了简化，预测的周期规定为一周。这样，问题变成了如果知道本周中心渔产（位置，产量和大小），预报下周（下下周）中心渔场（位置，产量和大小）。即使如此，问题也是相当困难的，因为渔场位置、大小是一种空间数据。同时，海况信息涉及 600 来个空间和非空间属性，回归的方法，决策树等方法并非很适合，我们采用了 CBR 这种 Lazy Learning 的方法[Aha 1997]。

在建造范例库的过程中。一个合理的、一致的范例表示方法是必不可少的，把情景(Situation)、解答(Solution)、结果(Outcome)用一种合适的方式表示出来是构造一个基于范例推理系统首要解决的问题。在本系统中，我们采用面向对象的技术表示海况和渔况，对象的每一个实例与后台数据库中的纪录对应，系统在运行中将动态地构造它们。

对积累的原始数据经过合适的预处理，才能将问题转化为适合机器学习的形式。原始的捕捞记录数据格式为捕捞时间、捕捞地点、产量、捕捞方式等（其中来源于个体渔业公司的部分数据，甚至连产量都没有）。我们必须在数据不完整的情况下，将它们聚集成中心渔场。同时，为了简便起见，聚集的中心渔场忽略其形状，只考虑其大小（相当于认为它是等面积的圆）。我们将原始捕捞纪录按时间（周）、捕捞方式分组，可以用类似于聚类的方法对原始的捕捞数据进行约简，形成代表性很强的中心渔场，然后用下述人机交互的方法修正：

1. 如果某个渔场的周围存在相邻的渔场，则将它们合并；
2. 重复步骤 2，直到没有新的合并可以进行；
3. 忽略产量低于一定阈值的区域（拒绝将其加入任何中心渔场）；

4. 如果合并后的渔场仍然很多，忽略一些面积和产量比较小的渔场；
5. 将合并以后的渔场作为中心渔场，并且中心渔场的位置为它们的几何中心，大小用包括的渔区数表示，产量为它们的累加。
6. 将步骤 6 的结果用可视化的方式提交给用户，让用户根据具体情况进一步调整；
7. 在数据库中保存修改后的结果。

利用我们开发的工具（如图 5.6），领域专家在很短的时间内就将历史数据整理完毕。

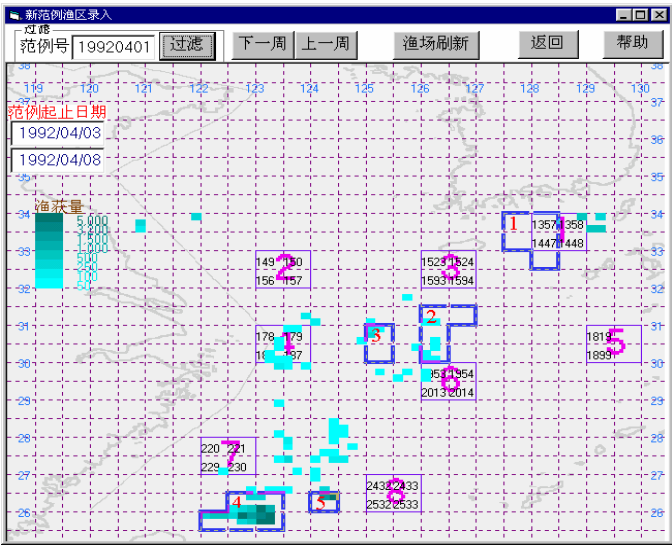


图 5.6 人机交互聚类得到中心渔场

5.13.2 相似性度量

基于范例推理中一个关键的环节是检索得到的相似范例应该尽可能的“好”。范例检索是在相似比较的基础上进行的，要检索到相似的范例来，完全要靠“什么程度才算相似”的定义了。如果定义有问题，检索的结果就不理想，也就谈不上应用的成功。因此相似度的定义十分重要。然而，相似性描述总是粗糙的、部分的、易变的。CASE 对于范例的优选通常遵循以下规则：尽可能的针对语义相似，结构相似，目标相似和个体相似四种约束，使用相应的计算方法，全面评估范例之间的相似性，然后针对各种相似性在问题域内的重要性，计算出总的动态加权相似性。

在我们的系统中，对空间条件属性，采用三个相似性度量方法（函数）：

- 基于渔场位置的相似

$$\text{sim1} = -\sum (W_i * \text{distance}(\text{pos}(\text{goal}) - \text{pos}(\text{source}))) / \sum w_i \quad (5.8)$$

- 基于温度场的相似

$$\text{sim2} = -\sum (w_i * \text{difference}(\text{temp}(\text{goal}) - \text{temp}(\text{source}))) / \sum w_i \quad (5.9)$$

- 基于温度梯度的相似

$$\text{sim3} = -\sum (w_i * \text{difference}(\text{delta}(\text{goal}) - \text{delta}(\text{source}))) / \sum w_i \quad (5.10)$$

其中， w_i 是权重。如果该温度测试点与样本的中心渔场距离 d_i 越近，权越大。计算方法为，如果距离 d_i 为 0，权重为 1；如果距离 d_i 为海洋区域的直径——最大距离 d_{\max} ，权值为一个小于 1 的数 w_0 （用户可以根据推理结果修正它）；其余插值。即有

$$w_i = 1 - \frac{d_i(1 - W_0)}{d_{\max}} \quad (5.11)$$

5.13.3 索引与检索

检索到的范例中含有与该范例相对应问题的解答，把该解答复用于当前的目标范例或新问题的求解，就是范例复用。复用的过程可以分为类比映射和类比转换。要使源范例的求解方案可能应用于目标范例，一个前提条件是，必须在源范例和目标范例的各个特征之间建立起一致的对应关系，并决定哪些关系和结构可映射到目标范例中。因此，在类比映射中，主要要解决的两个问题是：特征的辨识和对应，可映射关系和结构的选择。

范例的索引主要有近邻法、归纳法、知识导引法或这三种方法的结合。在本系统中，困难之处在于海况和渔况信息既是与空间相关的，又是与时间相关的。为表达清晰，我们作如下形式化。设 C_{t1} 为时刻 t_1 的海况，它是渔场发生的条件（条件属性）； G_{t1} 为时刻 t_1 的渔场（的位置，产量和大小，决策属性）；时刻 t_1 实例 $I_{t1} = (C_{t1}, G_{t1})$ 组成一个高维向量。 $I_{t1-tn} = \{I_{t1}, I_{t2}, \dots, I_{tn}\} = \{(C_{t1}, G_{t1}), (C_{t2}, G_{t2}), \dots, (C_{tn}, G_{tn})\}$ 构成一个时间上连续的实例序列，其中 t_2 是 t_1 的下一时刻（周）， t_n 是 t_1 的下 n 时刻（周）。 $\Delta G_i = G_{i+1} - G_i$ 反映了渔场的变化。

设 $\Gamma_{s'1-s'k}$ ， $\Gamma_{t'1-t'k}$ 是 I_{s1-sm} ， I_{t1-tn} 的两个长度为 k 的全序子集， $\Gamma_{s'1-s'k} \subseteq I_{s1-sm}$ ， $\Gamma_{t'1-t'k} \subseteq I_{t1-tn}$ ， $\text{sim}(\cdot)$ 是相似性度量的函数。如果 $\Gamma_{s'1-s'k} \cong \Gamma_{t'1-t'k}$ ，即

$$\forall I_{s'i} \in \Gamma_{s'1-s'k}, \forall I_{t'i} \in \Gamma_{t'1-t'k}, \text{sim}(I_{s'i}, I_{t'i}) \geq \delta \quad (1 \leq i \leq k), \text{ 其中 } \delta \text{ 是 } 0 \text{ 到 } 1 \text{ 之间的阈值,}$$

则称 I_{s1-sm} ， I_{t1-tn} 是 k - δ 完全相似的。显然， k 、 δ 越大， I_{s1-sm} ， I_{t1-tn} 的相似程度越高。

设 $C_{s'1-s'k}$ ， $C_{t'1-t'k}$ 是条件序列 $C_{s1-sm} = \{C_{s1}, \dots, C_{sm}\}$ ， $C_{t1-tn} = \{C_{t1}, \dots, C_{tn}\}$ 的两个长度为 k 的全序子集， $C_{s'1-s'k} \subseteq C_{s1-sm}$ ， $C_{t'1-t'k} \subseteq C_{t1-tn}$ 。如果 $C_{s'1-s'k} \cong C_{t'1-t'k}$ ，即

$$\forall C_{s'i} \in C_{s'1-s'k}, \forall C_{t'i} \in C_{t'1-t'k}, \text{sim}(C_{s'i}, C_{t'i}) \geq \delta \quad (1 \leq i \leq k), \text{ 其中 } \delta \text{ 是 } 0 \text{ 到 } 1 \text{ 之间的阈值,}$$

则称 I_{s1-sm} ， I_{t1-tn} 是 k - δ 条件相似的。

设 $I_{u-cur} = \{I_u, I_{u+1}, \dots, I_{cur}\}$ 表示当前渔场从时刻 u 到当前时刻 cur 长度为 $cur-u+1$ 的序列，则 $I_{u-cur+1} = \{I_u, I_{u+1}, \dots, I_{cur}, I_{cur+1}\} = \{I_{u-cur}, I_{cur+1}\}$ 表示当前渔场从时刻 u 到下一时刻 $cur+1$ 长度为 $cur-u+2$ 的发展过程，其中， $I_{cur+1} = (C_{cur+1}, G_{cur+1})$ ， C_{cur+1} 可以通过气象分析等方式比较准确地得到， G_{cur+1} 表示下周渔场，即为我们需要预测的内容。设 $\Gamma_{u'1-u'k}$ 为 $I_{u-cur+1}$ 的包含 I_{cur}, I_{cur+1} 全序子集，我们需要发现， $\Gamma_{u'1-u'k}$ 与某个历史渔场序列 I_{v-w} 的子序列 $\Gamma_{v'1-v'k}$ 存在 k - δ 完全相似。由 $\Gamma_{u'1-u'k} \cong \Gamma_{v'1-v'k}$ ，有海况相似 $C_{u'1-u'k} \cong C_{v'1-v'k}$ ，渔况相似 $G_{u'1-u'k} \cong G_{v'1-v'k}$ 。则有

$G_{cur+1} = G_{u'k} \cong G_{v'k}$ ，由 $G_{v'k}$ 可以得到 G_{cur+1} 的位置。

试验表明，完全相似的条件是非常强的。我们可以使用条件相似，设 $C'_{u'l-u'k}$ 为 $C_{u-cur+1}$ 的包含 C_{cur}, C_{cur+1} 全序条件子集，我们需要发现， $C'_{u'l-u'k}$ 与某个条件序列 C_{v-w} 的子序列 $C'_{v'l-v'k}$ 存在 $k-\delta$ 条件相似。由 $C'_{u'l-u'k} \cong C'_{v'l-v'k}$ ，有 $G_{cur+1} = G_{u'k} \cong G_{v'k}$ ，则 $G_{cur+1} = G_{cur} + \Delta G$ ， $\Delta G = G_{v'k} - G_{v'k-1}$ 为相似渔场中本周渔场到下周渔场的变化。这是我们在上一节中引入几个相似性函数的原因。

在实际系统中，由于 δ 是难以确定的。在相似性检索中，我们采用 k -近邻的方法，通过发现 k 个最相似的历史序列来计算下周渔场的位置，这样就不需要一个确定的 δ 值了，同时具有较强的鲁棒性。

5.13.4 基于框架的修正

框架是一种知识表示模式，1975 年由 Minsky 最早作为视觉感知、自然语言对话和其他复杂行为的基础提出[Minsky 1989]。对一类典型的实体，如状况、概念、事件等，以通用的数据结构的形式予以存储。当新的情况发生时，只要把新的数据加入这些数据结构，就形成了一个具体的实体。这样的关于典型实体的通用数据结构就称为框架(frame)。框架结构也提供了在一个具体上下文中对特定实体进行预见驱动的处理方式，在这个上下文环境下根据这种结构可以寻找那些预见的信息。框架结构中存放这些可预见信息的位置称为槽(slots)。一个框架由若干个槽组成，每个槽有它自己的名字，槽内的值描述框架所表示的实体的各个组成部分的种属性，每个槽的值又可由一个或多个侧面(facets)组成，各侧面从各个方面来描述槽的特性，每个侧面又有一个或多个侧面值，每个侧面值可以是一个或者一个概念的描述。框架理论，以其组织知识区块及处理预设知识上的简洁特性而被广泛应用，已成为重要的知识表示方法。然而既有的框架应用系统中，仍缺乏处理未确定式知识及预设知识冲突时的解决技术。

针对对海洋渔业资源预测研究，我们定义了一框架系统 KBIF，具备了表示不同重要性的未确定知识的能力。KBIF 的主要特性包括：权重—描述框架内各槽对框架的影响程度，信赖因素—描述槽本身的可信程度，及偏向因素—在次框架关系中描述继承的倾向程度。

鱼类的洄游规律受很多因素制约，变化非常复杂，难以用传统的数学方法和模型描述。基于范例推理的预测算法只能采用基于大量历史数据积累的部分因素进行预测，同时专家关于中心渔场规律的知识是不精确的，不完全的。对中心渔场的预测结果进行有效修正，就成为预测准确性的关键，因此系统采用三种知识处理模型：框架模型、黑板结构、模糊推理统一成为框架的知识表示，对中心渔场的预测结果加以修正，并将领域专家修正系统划分为三大部分：原子集、规则集、结论集，分别与三种模型相对应。三种模型的应用使专家预测系统适应了可扩展性、规则定义的灵活性的基本原则。

模型一：框架模型 KBIF。框架模型的知识表示对多种制约因素进行规则表示，使领域专家可以依据框架结构，灵活设定制约因素，各因素间通过不断沟通，形成一个具有一致性、整体性的系统，并以它引导各元素互相合作达成目标。

修正系统原子集的构造方法如下：首先，对中心渔场形成而受到的多种因素的制约加以细分，分解形成修正规则的最基本元素，每个因素成为预测规则的一个原子，而此类基本元素的全体则称为原子集，即框架。原子集利用 KBIF 的知识表示原理，定义了预测算法的所有可表示的制约因素，其中每一因素可以作为框架的槽。框架整体命名为“修正系统规则集”，

并将框架分解为三层，分别包括：台站数据、长江径流量、流系等。台站数据包括了：风速、风向、降水，气温、气压、时间、水温、盐度、站名等因素。长江径流量包括：时间、月平均。流系包括：表层水温、表层水温梯度、表层盐度、表层盐度梯度、垂直水温梯度、垂直盐度梯度、底层水温、底层水温梯度、底层盐度、底层盐度梯度、时间、叶绿素。原子集序列可以根据领域专家对制约因素的深入了解不断扩展，适应了中心渔场的预测扩展性的要求。

模型二：黑板结构。我们在 KBIF 基础上建立了广义式黑板结构子系统。藉由此子系统，使用者可任意地经同质性及异质性切割将解题空间规划成阶层式子结构区即规则集，此子系统则能在广义式黑板系统中支持决策概念。

我们定义了虚黑板架构，并依特定领域即渔业资源预测的特性，将之转化为黑板系统。它整合了知识源即原子集、规则集生成器和模糊推理的具体应用。系统结构设计成一微型黑板架构，其中包含了一组知识原子集、一个微黑板及一个规则生成控制器。这种结构将知识认识层与经验推理层有效的结合为一体。其中知识原子集隐含在系统中，与框架模型相对应，定义了模型中的所有原子集，并可根据需求加以扩展；规则生成控制器可以把知识原子集按照专家提供条件进行组合，逐一列入黑板，并在生成器中提供了区间与枚举两种功能来增进推论的效率，最终产生出特定规则。这种方式提供了渐进地建立知识规则能力；也减轻了专家经验推理的负荷，避免因着重于知识的内部结构，而忽略了所表达的领域知识。系统使用的图形界面来表达这样的结构，同时也增强了知识表示法和有效的推论能力以解决问题。

模型三：模糊推理。用模糊推理的方法将规则运用在范例推理得到的结果集上，对中心渔场的预测进行合理的修正，从而使预测结果更准确有效。

在专家预测修正系统的规则结论集合中，对中心渔场的位置（X 轴，Y 轴）向、渔场大小、渔产量大小应用模糊定义，将 CBR 预测出的渔场数据通过向量修正，得到新的预测数据。并加入了权序列模式，即定义出每一结论的可信度，对相似条件下的结论按可信度进行排列。在预测结束后可以对专家结论集进行修正，达到知识学习的目的，使问题不断逼近可解。

5.13.5 实验结果

系统的结构如图 5.6 所示，历史数据经过适当的聚类、变换等预处理，保存在海况、渔况范例库（数据库）中。为了提高预测精度，我们采用了多策略的相似检索，利用不同的相似性计算方法找出邻近的相似的渔场序列。然后，利用领域专家编辑的、已经保存在系统中的专家规则对预报结果进行修正。机器学习的知识与人类专家的知识结合起来使用，将进一步提高预报的精度和系统的可靠性。预测冬天的渔场时，我们就可以不考虑夏天的历史序列；离渔场位置较远的海况对中心渔场的发展影响不大，我们可以只选择那些离中心渔场较近的海况，从 600 来个海况属性动态地选取 150 个左右。这样，在保障预测精度的前提下，通过时间、位置过滤，大大地减少计算量，系统响应控制在 7 秒钟左右。可视化的人机交互界面，为中心渔场的输入、预报结果的输出、海况信息的检查、领域专家的修正提供了很好的接口（图 5.7）。

我们采用报全率和报准率衡量预测的结果，报准率是指预报渔场与实际渔场中心位置的差别，预报的中心与实际的中心相差不到一个渔区，渔业专家则认为预测精度达到 80%。报全率是指预测渔场与实际渔场重叠的比例。我们的回顾性预报（回报）表明，系统的平均预报精度达到 78%，基本具备了应用推广的价值。目前我们已经完成了示范试验，正在与上海渔政局合作，着手应用到实际生产预报中。

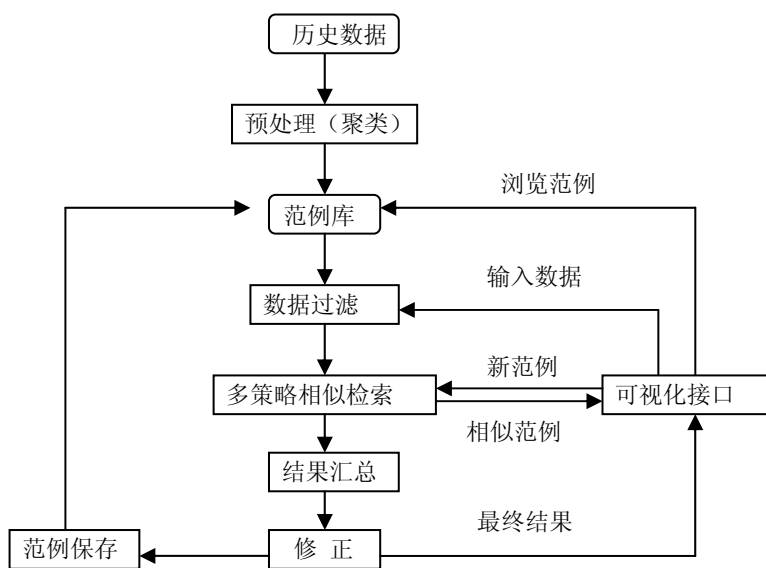


图5.7 系统的结构

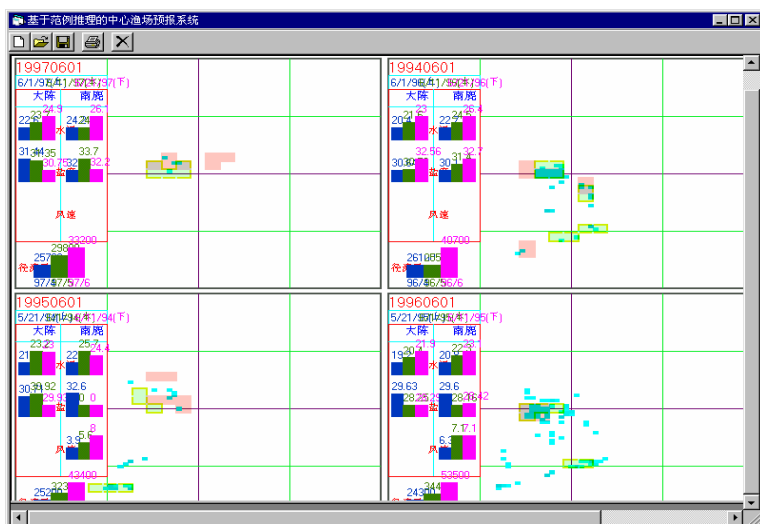


图 5.8 海况、渔场布图

习题

1. 用自己的语言描述什么是基于范例推理，简述几个 CBR 系统。
2. 描述范例的表示形式以及基于范例推理的基本思想。
3. 阐述基于范例推理的适用场合，并通过流程图描述基于范例学习的一般过程。
4. 与基于规则的推理（RBR）相比较，CBR 有哪些优点？CBR（CBL）系统的学习能力体现在哪里。
5. 简述目标范例和源范例的几种相似性，并比较几种常用相似性计算距离度量方法的特点。
6. 查阅相关资料介绍一个基于范例推理的应用系统，并讨论 CBR 系统中包含哪些关键问题，在应用系统中是如何实现的。
7. 用具体的系统说明基于范例推理与专家系统有什么区别。

第六章 概率推理

6.1 概述

贝叶斯网络是用来表示变量间连接概率的图形模式，它提供了一种自然的表示因果信息的方法，用来发现数据间的潜在关系。在这个网络中，用节点表示变量，有向边表示变量间的依赖关系。贝叶斯理论给出了信任函数在数学上的计算方法，具有稳固的数学基础，同时它刻画了信任度与证据的一致性及其信任度随证据而变化的增量学习特性；在数据挖掘中，贝叶斯网络可以处理不完整和带有噪声的数据集，它用概率测度的权重来描述数据间的相关性，从而解决了数据间的不一致性，甚至是相互独立的问题；用图形的方法描述数据间的相互关系，语义清晰、可理解性强，这有助于利用数据间的因果关系进行预测分析。贝叶斯方法正在以其独特的不确定性知识表达形式、丰富的概率表达能力、综合先验知识的增量学习特性等成为当前数据挖掘众多方法中最为引人注目的焦点之一。

6.1.1 贝叶斯网络的发展历史

贝叶斯（Reverend Thomas Bayes 1702-1761）学派奠基性的工作是贝叶斯的论文“关于几率性问题求解的评论”。或许是他自己感觉到它的学说还有不完善的地方，这一论文在他生前并没有发表，而是在他死后，由他的朋友发表的。著名的数学家拉普拉斯（Laplace P. S.）用贝叶斯的方法导出了重要的“相继律”，贝叶斯的方法和理论逐渐被人理解和重视起来。但由于当时贝叶斯方法在理论和实际应用中还存在很多不完善的地方，因而在十九世纪并未被普遍接受。二十世纪初，意大利的菲纳特（B. de Finetti）以及英国的杰弗莱（Jeffreys H.）都对贝叶斯学派的理论作出重要的贡献。第二次世界大战后，瓦尔德（Wald A.）提出了统计的决策理论，在这一理论中，贝叶斯解占有重要的地位；信息论的发展也对贝叶斯学派做出了新的贡献。1958 年英国最悠久的统计杂志 *Biometrika* 全文重新刊登了贝叶斯的论文，20 世纪 50 年代，以罗宾斯（Robbins H.）为代表，提出了经验贝叶斯方法和经典方法相结合，引起统计界的广泛注意，这一方法很快就显示出它的优点，成为很活跃的一个方向。

随着人工智能的发展，尤其是机器学习、数据挖掘等兴起，为贝叶斯理论的发展和应用提供了更为广阔的空间。贝叶斯理论的内涵也比以前有了很大的变化。80 年代贝叶斯网络用于专家系统的知识表示，90 年代进一步研究可学习的贝叶斯网络，用于数据采掘和机器学习。近年来，贝叶斯学习理论方面的文章更是层出不穷，内容涵盖了人工智能的大部分领域，包括因果推理、不确定性知识表达、模式识别和聚类分析等。并且出现了专门研究贝叶斯理论的组织 and 学术刊物 ISBA。

6.1.2 贝叶斯方法的基本观点

贝叶斯分析方法的特点是使用概率去表示所有形式的不确定性，学习或其它形式的推理都用概率规则来实现。贝叶斯学习的结果表示为随机变量的概率分布，它可以解释为我们对不同可能性的信任程度。贝叶斯学派的起点是贝叶斯的两项工作：贝叶斯定理和贝叶斯假设。

贝叶斯定理将事件的先验概率与后验概率联系起来。假定随机向量 x, θ 的联合分布密度是

$p(x, \theta)$ ，它们的边际密度分别为 $p(x), p(\theta)$ 。一般情况下设 x 是观测向量， θ 是未知参数向量，通过观测向量获得未知参数向量的估计，贝叶斯定理记作：

$$p(\theta | x) = \frac{\pi(\theta) * p(x | \theta)}{p(x)} = \frac{\pi(\theta) * p(x | \theta)}{\int \pi(\theta) * p(x | \theta) d\theta} \quad (\pi(\theta) \text{ 是 } \theta \text{ 的先验分布}) \quad (6.1)$$

从上式我们可以看出，对未知参数向量的估计综合了它的先验信息和样本信息，而传统的参数估计方法只从样本数据获取信息如最大似然估计。贝叶斯方法对未知参数向量估计的一般过程为：

1. 将未知参数看成是随机向量。这是贝叶斯方法与传统的参数估计方法的最大区别。
2. 根据以往对参数 θ 的知识，确定先验分布 $\pi(\theta)$ ，它是贝叶斯方法容易引起争议的一步，因此而受到经典统计界的攻击。
3. 计算后验分布密度，做出对未知参数的推断。

在第二步，如果没有任何以往的知识来帮助确定 $\pi(\theta)$ ，贝叶斯提出可以采用均匀分布作为其分布，即参数在它的变化范围内，取到各个值的机会是相同的，称这个假定为贝叶斯假设。贝叶斯假设在直觉上易于被人们所接受，然而它在处理无信息先验分布，尤其是未知参数无界的情况却遇到了困难。经验贝叶斯估计 EB (Empirical Bayes Estimator) 把经典的方法和贝叶斯方法结合在一起，用经典的方法获得样本的边际密度 $p(x)$ ，然后通过下式来确定先验分布 $\pi(\theta)$ ：

$$p(x) = \int_{-\infty}^{+\infty} \pi(\theta) p(x | \theta) d\theta$$

6.1.3 贝叶斯网络在数据挖掘中的应用

1. 贝叶斯方法用于分类及回归分析

分类规则发现是根据客体的特征向量值及其它约束条件将其分到某个类别中。在数据挖掘中，主要研究如何从数据或经验中学习这些分类规则。对于分类问题，有些情况下，输入特征向量唯一对应着一个类别，称它为确定性的分类问题；而有些情况下，则会出现类别的重叠现象，也就是说，来自于不同类别的样本从外观特征上具有极大的相似性，这时我们只能说某一样本属于某一类别的概率是多大，然而我们却必须为它选择一个类别。贝叶斯学派采用两种方法处理这种情况：一是选择后验概率最大的类别；二是选择效用函数最大（或损失最小）的类别。设特征向量 $X = (x_1, x_2 \cdots x_m)$ ，类别向量 $C = (c_1, c_2 \cdots c_l)$ 。分类的目的是把特征向量 X 归入到某个类别 $c_i (i \in (1 \cdots l))$ 中，第一种方法选择后验概率最大的类别，即：

$P(c_i | x) \geq P(c_j | x) \quad j \in (1 \cdots l)$ ，此时取判别函数： $r_i(x) = P(c_i | x)$ 。可以证明，这种方

法能够保证分类误差最小。

第二种方法，在决策理论中经常用到，它采用平均效益的大小来衡量决策风险的大小，这实际上与不确定性的程度密切相关。设把属于类别 c_i 的特征向量 X 错误地划分到类别 c_j 中

的损失为 $L_{ij}(X)$ ，它选择损失最小的类别，即：
$$\text{Minimize} \left\{ \sum_{j=1}^l L_{ij}(x) \cdot P(c_j | x) \right\}$$
，此时取判

别函数：
$$r_i(x) = \sum_{j=1}^l L_{ij}(x) \cdot P(c_j | x)$$
。当 $L_{ij}(X)$ 的对角线元素取为 0，非对角线元素取为 1，

即正确分类的损失为 0，错误分类的损失都相同时，风险损失最小与后验概率最大是等价的。

在数据挖掘中，对贝叶斯分类的研究主要集中在如何从数据中学习特征向量的分布、特征向量间的相关性等来确定最好的 $P(c_i | x)$ 和 $L_{ij}(X)$ 。现已具有求解它们的成功模型，如单贝叶斯 (Naïve Bayesian)、贝叶斯网络 (Bayesian Network) 及贝叶斯神经网络 (Bayesian Neural Network) 等。目前，贝叶斯分类方法已在文本分类、字母识别、经济预测等领域获得了成功的应用。

2. 用于因果推理和不确定知识表达

贝叶斯网络是随机变量间的概率关系的图表示。近年来，贝叶斯网络已经成为专家系统中不确定性知识的主要表示方法，并且涌现出大批的从数据学习贝叶斯网络的算法 [Cooper 92]。这些技术在数据建模、不确定性推理等方面已取得了相当的成功。

与数据挖掘中的其它知识表示的方法如：规则表示、决策树、人工神经网络等相比，贝叶斯网络在知识表示方面具有下列优点 [Cooper 1992]：

- 贝叶斯网络能够方便的处理不完全数据。例如考虑具有相关关系的多个输入变量的分类或回归问题，对标准的监督学习算法而言，变量间的相关性并不是它们处理的关键因素，当这些变量中有某个缺值时，它们的预测结果就会出现很大的偏差。而贝叶斯网络则提供了较为直观的概率关联关系。
- 贝叶斯网络能够学习变量间的因果关系。因果关系是数据挖掘中的极为重要的模式。原因有二：在数据分析中，因果关系有利于对领域知识的理解；在干扰较多时，便于作出精确的预测。例如市场销售分析人员想知道增加广告投入是否能提高产品的销量。为回答这个问题，分析人员必须知道，在某种程度上，广告投入是否是提高销量的原因。即使没有这方面的实验数据，贝叶斯网络对这类问题的回答也是相当简单的，因为这种因果关系已经包含在贝叶斯网络中了。
- 贝叶斯网络与贝叶斯统计相结合能够充分利用领域知识和样本数据的信息。任何从事过实际建模任务的人都会知道，先验信息或领域知识在建模方面的重要性，尤其是在样本数据稀疏或数据较难获得的时候，一些商业方面的专家系统完全根据领域专家知识来构建就是一个很好的例证。贝叶斯网络用弧表示变量间的依赖关系，用概率分布表来表示依赖关系的强弱，将先验信息与样本知识有机结合起来。
- 贝叶斯方法与其它模型相结合，有效的避免了数据的过分拟合问题。

3. 用于聚类模式发现

广义上讲，聚类问题是一种特殊的模型选择问题，每个聚类模式都可以看作是一种模型。聚类的任务就是从这些众多的模型中，通过一定的分析、综合策略，选择出体现数据本质的模式。贝叶斯方法通过综合先验的模型知识和当前的数据特点，来实现选择最优模型的目的。

Vaithyanathan 等提出了利用贝叶斯分析的基于模型的层次聚类方法 [Vaithyanathan 98]。它

通过特征集的划分，将数据组织成层次结构。这些特征或者在每个类别中都有唯一的分布，或者在某些类别中具有共同的分布。它同时给出了利用边界似然来确定模型结构的方法，包括自动决定类别的个数、模型树的深度和每个类别的特征子集。

AutoClass 是利用贝叶斯方法实现聚类的一个典型系统。该系统通过搜索模型空间所有的分类可能性，来自动决定分类类别的个数和模型描述的复杂性。它允许在一定的类别内属性之间具有一定的相关性，各个类之间具有一定的继承性（在类层次结构中，某些类共享一定的模型参数）。读者可从网页 <http://ic-www.arc.nasa.gov/ic/projects/bayes-group/autoclass> 看到有关 AutoClass 的信息。

以上仅仅给出贝叶斯方法的几个典型应用，然而贝叶斯方法在数据挖掘中的应用远不止这些，比如贝叶斯方法与神经网络相结合的贝叶斯神经网络，贝叶斯方法与统计学习相结合的贝叶斯点机等。有兴趣的读者可参考相关书籍[Amari 85]。

6.2 贝叶斯概率基础

6.2.1 概率论基础

概率论是研究随机现象规律性的数学。随机现象是指在相同的条件下，其出现的结果是不确定的现象。随机现象又可分为个别随机现象和大量的随机现象。对大量随机现象进行观察所得到的规律性，被人们称为统计规律性。

在统计上，我们习惯把一次对现象的观察、登记或实验叫做一次试验。随机性试验是指对随机现象的观察。随机试验在完全相同的条件下，可能出现不同的结果，但所有可能结果的范围是可以估计的，即随机试验的结果具有不确定性和可预计性。在统计上，一般把随机试验的结果，即随机现象的具体表现叫做随机事件，简称为事件。

随机事件是指试验中可能出现，也可能不出现的结果。在随机现象中，某标志表现的频数是指全部观察值(即样本)中拥有该标志表现的单位总数。

例 6.1 我们为研究某工厂的产品质量情况，采用随机抽样检查，各次检查(试验)的数量(即样本容量)不同，其结果记录如表 6.1 所示。

表 6.1 各次质量抽查结果

抽查件数	5	10	50	100	300	600	1000	5000	10000
合格频数	5	8	44	91	272	542	899	4510	8999
合格频率	1	0.8	0.88	0.91	0.907	0.892	0.899	0.902	0.8997

表中抽查件数是指每次试验(每个样本)的单位总数，这只是为说明问题设计的，而现实统计中多数情况下只有一个样本。表中合格频数是指合格品这种标志表现在各样本中的单位总数。表中的合格频率是指各样本的合格频数占样本单位总数(样本容量)的比重，即相对频数。从表中各数值的观察中可以看出频数与频率之间的关系，同时也会发现其中蕴含着一种统计规律性。就是随着样本单位数的扩大，其合格率稳定地趋向 0.9 左右。即合格频率总是围绕着一个固定常数 $p=0.9$ 摆动。可见 p 是这一系列试验的统计稳定中心，它可以表明一次检查中合格品出现的可能性，即概率。

定义 6.1 统计概率：若在大量重复试验中，事件 A 发生的频率稳定地接近于一个固定的

常数 p , 它表明事件 A 出现的可能性大小, 则称此常数 p 为事件 A 发生的概率, 记为 $P(A)$, 即

$$p=P(A) \quad (6.1)$$

可见概率就是频率的稳定中心。任何事件 A 的概率为不大于 1 的非负实数, 即

$$0 \leq P(A) \leq 1$$

概率的统计定义与频率联系紧密易于理解, 但是用试验的方法来求解概率是很麻烦的, 有时甚至是不可能的。因此我们常用古典概率和几何概率来解决概率的计算等问题。

定义 6.2 古典概率: 设一种试验, 有且仅有有限的 N 个等可能结果, 即 N 个基本事件, 而 A 事件包含着其中的 K 个可能结果, 则称 K/N 为事件 A 的概率, 记为 $P(A)$ 。

$$p(A) = \frac{K}{N} \quad (6.2)$$

古典概率的计算需要知道全部的基本事件数目, 它局限于离散的有限总体。而对无限总体或全部基本事件未知的情况之下, 求解概率需采用几何概型, 同时几何概型也提供了概率的一般性定义。

几何型随机试验: 假设 Ω 是 M 维空间中的一个有界区域, 现以 $L(\Omega)$ 表示 Ω 的体积, 考虑随机试验是向 Ω 内均匀地投掷一随机点, 并假设: (1) 随机点可能落到区域 Ω 内的任何一处, 但不可能落在 Ω 之外。(2) 随机点在 Ω 中分布均匀, 即落入 Ω 中任何区域的可能性与该区域之 M 维体积成正比, 而与区域的形状及在 Ω 中的位置无关。在上述条件下我们称试验为几何型随机试验; Ω 是基本事件空间。

几何型随机试验中的事件: 假设 Ω 是几何型随机试验的基本事件空间, A 是 Ω 中可以用体积来度量的子集, $L(A)$ 是 A 的 M 维体积。那么“随机点落入区域 A 内”的事件以 A 表示。在 Ω 中能用体积度量的子集叫可测集, 每一个可测集都可视为一个事件, 一切可测子集的集合以 F 来表示。

定义 6.3 几何概率: 假设 Ω 是几何型随机试验的基本事件空间, F 是 Ω 中一切可测集的集合, 则对于 F 中的任意事件 A 的概率 $P(A)$ 为 A 与 Ω 的体积之比, 即

$$p(A) = \frac{V(A)}{V(\Omega)} \quad (6.3)$$

定义 6.4 条件概率: 我们把事件 B 已经出现的条件下, 事件 A 发生的概率记做为 $P(A|B)$ 。并称之为在 B 出现的条件下 A 出现的条件概率, 而称 $P(A)$ 为无条件概率。

例 6.2 袋子中有两个白色球和一个黑球, 现依次从袋子中取出两个球, 试问: (1) 第一次取出白球的概率? (2) 已知第一次取得白球的条件下, 第二次仍取得白球的概率?

解: 设 A 为第一次取得白球的事件, B 为第二次取得白球的事件, 则 $\{B|A\}$ 为第一次取得白球的条件下, 第二次仍取得白球的事件。根据定义有:

(1) 不论是重复抽取还是不重复抽取, $P(A) = 2/3$

(2) 在不重复抽取时: $P(B|A) = 1/2$

(3) 在可重复抽取时: $P(B|A) = P(B) = 2/3$, 即相当于无条件概率。

若事件 A 与 B 中的任一个出现, 并不影响另一事件出现的概率, 即当 $P(A) = P(A \cdot B)$ 或 $P(B) = P(B \cdot A)$ 时, 则称 A 与 B 是相互独立的事件。

定理 6.1 加法定理

两个不相容(互斥)事件之和的概率, 等于两个事件概率之和, 即

$$P(A+B) = P(A) + P(B)$$

两个互逆事件 A 和 A^{-1} 的概率之和为 1。即当 $A + A^{-1} = \Omega$, 且 A 与 A^{-1} 互斥, 则 $P(A) + P(A^{-1}) = 1$, 或常有 $P(A) = 1 - P(A^{-1})$ 。

若 A 、 B 为两任意事件, 则:

$$P(A+B) = P(A) + P(B) - P(AB)$$

成立。此定理可推广到三个以上事件的情形：

$$P(A+B+C) = P(A) + P(B) + P(C) - P(AB) - P(BC) - P(CA) + P(ABC)。$$

定理 6.2 乘法定理

设 A、B 为两个不相容(互斥)的非零事件，则其乘积的概率等于 A 和 B 概率的乘积。即：

$$P(A \cdot B) = P(A) \cdot P(B) \text{ 或 } P(A \cdot B) = P(B) \cdot P(A)$$

设 A、B 为两个任意的非零事件，则其乘积的概率等于 A(或 B)的概率与在 A(或 B)出现的条件下 B(或 A)出现的条件概率的乘积。

$$P(A \cdot B) = P(A) \cdot P(B|A) \text{ 或 } P(A \cdot B) = P(B) \cdot P(A|B)$$

此定理可以推广到三个以上事件的乘积情形，即当 n 个事件的乘积 $P(A_1 A_2 \dots A_{n-1}) > 0$ 时，则乘积的概率为：

$$P(A_1 A_2 \dots A_n) = P(A_1) \cdot P(A_2 | A_1) \cdot P(A_3 | A_1 A_2) \dots P(A_n | A_1 A_2 A_{n-1})$$

当事件两两独立时，则有：

$$P(A_1 A_2 \dots A_n) = P(A_1) \cdot P(A_2) \cdot P(A_3) \dots P(A_n)$$

6.2.2 贝叶斯概率

1. 先验概率。先验概率是指根据历史的资料或主观判断所确定的各事件发生的概率，该类概率没能经过实验证实，属于检验前的概率，所以称之为先验概率。先验概率一般分为两类，一是客观先验概率，是指利用过去的历史资料计算得到的概率；二是主观先验概率，是指在无历史资料或历史资料不全的时候，只能凭借人们的主观经验来判断取得的概率。

2. 后验概率。后验概率一般是指利用贝叶斯公式，结合调查等方式获取了新的附加信息，对先验概率进行修正后得到的更符合实际的概率。

3. 联合概率。联合概率也叫乘法公式，是指两个任意事件的乘积的概率，或称之为交事件的概率。

4. 全概率公式。如果影响事件 A 的所有因素 B_1, B_2, \dots 满足： $B_i \cdot B_j = \Phi, (i \neq j)$

且： $P(\bigcup B_i) = 1, P(B_i) > 0, i = 1, 2, \dots$

$$\text{则必有： } P(A) = \sum P(B_i)P(A | B_i) \quad (6.4)$$

5 贝叶斯公式

贝叶斯公式，也叫后验概率公式，还叫逆概率公式，其用途很广。

设：先验概率为 $P(B_i)$

调查所获的新附加信息为 $P(A_j | B_i)$

$i=1, 2, \dots, n$

$j=1, 2, \dots, m$

则贝叶斯公式计算的后验概率为：

$$P(B_i | A_j) = \frac{P(B_i)P(A_j | B_i)}{\sum_{k=1}^m P(B_i)P(A_k | B_i)} \quad (6.5)$$

例 6.3 某厂生产的某种产品，由三个生产小组都生产两种规格的该产品，其有关产量资料如表 6.2 所示：

表 6.2 三个小组的日产量 单位：件

第一组 A1	2 000	1 000	3 000
第二组 A2	1 500	500	2 000
第三组 A3	500	500	1 000
合 计	4 000	2 000	6 000

现从这 6000 件产品中随机抽取一件，完成如下各问。

1. 用古典概率计算下列概率：

1) 分别计算该产品是第一、二、三组生产的概率

解： $P(A1) = 3000/6000 = 1/2$

$P(A2) = 2000/6000 = 1/3$

$P(A3) = 1000/6000 = 1/6$

2) 分别计算该产品为哪一型号的概率？

解： $P(B1) = 4000/6000 = 2/3$

$P(B2) = 2000/6000 = 1/3$

3) 计算该产品既是第一组生产的又是 B1 型的概率

解： $P(A1 \cdot B1) = 2000/6000 = 1/3$

4) 若已知该产品为第一组生产的，问其为 B1 型产品的概率？

解： $P(B1|A1) = 2000/3000 = 2/3$

5) 若发现该产品是 B2 型产品，求其由第一、二、三组生产的概率分别为多少？

解： $P(A1|B2) = 1000/2000 = 1/2$

$P(A2|B2) = 500/2000 = 1/4$

$P(A3|B2) = 500/2000 = 1/4$

2. 用条件概率计算

1) 若已知该产品为第一组生产的，问其为 B1 型产品的概率？

$P(B1|A1) = (1/3) / (1/2) = 2/3$

2) 若发现该产品是 B2 型产品，求其由第一、二、三组生产的概率分别为多少？

$P(A1|B2) = (1/6) / (1/3) = 1/2$

$P(A2|B2) = (1/12) / (1/3) = 1/4$

$P(A3|B2) = (1/12) / (1/3) = 1/4$

3. 用贝叶斯后验概率公式分别计算如下两问。

1) 已知： $P(B1) = 4000/6000 = 2/3$

$P(B2) = 2000/6000 = 1/3$

$P(A1|B1) = 1/2$

$P(A1|B2) = 1/2$

求：当发现抽出的该产品是第一组 A1 生产的，问该产品是 B2 型概率为多少？

解：计算各联合概率有：

$P(B1)P(A1|B1) = (2/3)(1/2) = 1/3$

$P(B2)P(A1|B2) = (1/3)(1/2) = 1/6$

计算全概率有： $P(A1) = (1/3) + (1/6) = 1/2$

则用贝叶斯公式有： $P(B2|A1) = (1/6) \div (1/2) = 1/3$

2) 已知： $P(A1) = 3 000/6 000 = 1/2$

$P(A2) = 2 000/6 000 = 1/3$

$P(A3) = 1 000/6 000 = 1/6$

$$P(B2|A1) = 1000/3000 = 1/3$$

$$P(B2|A2) = 500/2000 = 1/4$$

$$P(B2|A3) = 500/1000 = 1/2$$

求：当发现该产品为 B2 型的，问其由第一、二、三组生产的概率各为多少？

解：计算联合概率有：

$$P(A1)P(B2|A1) = (1/2)(1/3) = 1/6$$

$$P(A2)P(B2|A2) = (1/3)(1/4) = 1/12$$

$$P(A3)P(B2|A3) = (1/6)(1/2) = 1/12$$

计算全概率 $P(B2)$ 为：

$$\begin{aligned} P(B2) &= \sum P(Ai)P(B2|Ai) \\ &= (1/2)(1/3) + (1/3)(1/4) + (1/6)(1/2) = 1/3 \end{aligned}$$

利用贝叶斯公式计算后验概率分别为：

$$P(A1|B2) = (1/6) \div (1/3) = 1/2$$

$$P(A2|B2) = (1/12) \div (1/3) = 1/4$$

$$P(A3|B2) = (1/12) \div (1/3) = 1/4$$

6.3 贝叶斯问题求解

贝叶斯学习理论利用先验信息和样本数据来获得对未知样本的估计，而概率（联合概率和条件概率）是先验信息和样本数据信息在贝叶斯学习理论中的表现形式。如何获得这些概率（也称之为密度估计）是贝叶斯学习理论争议较多的地方。贝叶斯密度估计研究如何根据样本的数据信息和人类专家的先验知识获得对未知变量（向量）的分布及其参数的估计。它有两个过程：一是确定未知变量的先验分布；一是获得相应分布的参数估计。如果以前对所有信息一无所知，称这种分布为无信息先验分布；如果知道其分布求它的分布参数，称之为有信息先验分布。由于在数据挖掘中，从数据中学习是它的最基本特性，所以无信息先验分布是贝叶斯学习理论的主要研究对象。

选取贝叶斯先验概率是用贝叶斯模型求解的第一步，也是比较关键的一步。常用的选取先验分布的方法有主观和客观两种。主观的方法是借助人的经验、专家的知识等来指定其先验概率。而客观的方法是通过直接分析数据的特点，来观察数据变化的统计特征，它要求有足够多的数据才能真正体现数据的真实分布。在实际应用中，这两种方法经常是结合在一起使用的。下面给出几种常用的先验概率的选取方法，为此我们首先给出几个定义。

令 θ 表示模型的参数， $X = (x_1, x_2, \dots, x_n)$ 表示观测数据， $\pi(\theta)$ 是参数的先验分布，它表示在没有任何证据出现的情况下，对参数的信任程度。

$l(x_1, x_2, \dots, x_n | \theta) \propto p(x_1, x_2, \dots, x_n | \theta)$ 是似然函数，它表示在已知参数的情况下，对未知样本的信任程度。 $h(\theta | x_1, x_2, \dots, x_n) \propto p(\theta | x_1, x_2, \dots, x_n)$ 表示观测到新证据后，对参数信任程度的改变。贝叶斯定理清晰地表示出了它们之间的关系：

$$h(\theta | x_1, x_2, \dots, x_n) = \frac{\pi(\theta)p(x_1, x_2, \dots, x_n | \theta)}{\int \pi(\theta)p(x_1, x_2, \dots, x_n | \theta)d\theta} \propto \pi(\theta)l(x_1, x_2, \dots, x_n | \theta) \quad (6.6)$$

定义 6.5 分布密度的核：如果随机变量 z 的分布密度 $f(x)$ 可以分解为 $f(x) = cg(x)$ ，其中 c 是与 x 无关的常量，称 $g(x)$ 是 $f(x)$ 的核，记为 $f(x) \propto g(x)$ 。只要知道了分布密度的核，利用分布密度在全空间上的积分为 1，就可以确定相应的常数。因此，要求随机变量的分布密度，关键是求出分布密度的核。

定义 6.6 充分统计量：对参数 θ 而言，统计量 $t(x_1, x_2 \cdots x_n)$ 称为充分的，如果不论 θ 的先验分布是什么，相应的后验分布 $h(\theta | x_1, x_2 \cdots x_n)$ 总是 θ 和 $t(x_1, x_2 \cdots x_n)$ 的函数。

这一定义明确地表示了样本中关于 θ 的信息均可由它的充分统计量来反映，因此后验分布通过统计量与样本发生联系。下面我们给出判断一个统计量是否为充分统计量的定理，即著名的奈曼因子分解定理：

定理 6.3 若样本 $x_1, x_2 \cdots x_n$ 对参数 θ 的条件密度能表示成 $g(x_1, x_2 \cdots x_n)$ 与 $f(\theta, t(x_1, x_2 \cdots x_n))$ 的乘积，则 $t(x_1, x_2 \cdots x_n)$ 对参数是充分的，反之亦真。

6.3.1 几种常用的先验分布选取方法

1. 共轭分布族

Raiffa 和 Schaifeer 提出先验分布应选取共轭分布，即要求后验分布与先验分布属于同一分布类型。它的一般描述为：

定义 6.7 设样本 X_1, X_2, \dots, X_n 对参数 θ 的条件分布为 $p(x_1, x_2, \dots, x_n | \theta)$ ，如果先验分布密度函数 $\pi(\theta)$ 决定的后验密度 $\pi(\theta | x)$ 与 $\pi(\theta)$ 同属于一种类型，则称 $\pi(\theta)$ 为 $p(x | \theta)$ 的共轭分布。

定义 6.8 设 $P = \{ p(x | \theta) : \theta \in \Theta \}$ 是以 θ 为参数的密度函数族， $H = \{ \pi(\theta) \}$ 是 θ 的先验分布族，假设对任何 $p \in P$ 和 $\pi \in H$ ，得到的后验分布 $\pi(\theta | x)$ 仍然在 H 族中，则称 H 为 P 的共轭分布族。

因为当样本分布与先验分布的密度函数都是 θ 的指数函数时，它们相乘后指数相加，结果仍是同一类型的指数函数，只相差一个常数比例因子。所以有如下定理：

定理 6.4 如果随机变量 Z 的分布密度函数 $f(x)$ 的核为指数函数，则该分布属于共轭分布族。

核为指数函数的分布构成指数函数族。指数函数族包括二项分布、多项分布、正态分布、Gamma 分布、Poisson 分布和多变量正态分布等。它们都是共轭分布。常用的共轭分布还有 Dirichlet 分布。

用共轭分布作先验可以将历史上做过的各次试验进行合理综合，也可以为今后的试验结果分析提供一个合理的前提。由于非共轭分布的计算实际上是相当困难的，相比之下，共轭分布计算后验只需要利用先验做乘法，其计算特别简单。可以说共轭分布族为贝叶斯学习的实际使用铺平了道路。

2. 最大熵原则

熵是信息论中描述事物不确定性的程度的一个概念。如果一个随机变量 x 只取 a 与 b 两个

不同的值，比较下面两种情况：

$$(1) \quad p(x=a)=0.98, \quad p(x=a)=0.02$$

$$(2) \quad p(x=a)=0.45, \quad p(x=a)=0.55$$

很明显，(1)的不确定性要比(2)的不确定性小得多，而且从直觉上也可以看得出当取的两个值得概率相等时，不确定性达到最大。

定义 6.9 设随机变量 x 是离散的，它取 $a_1, a_2 \cdots a_k \cdots$ 至多可列个值，且 $p(x=a_i)=p_i$ ， $i=1,2,\cdots$ ，则 $H(x)=-\sum_i p_i \ln p_i$ 称为 x 的熵。对连续型随机变量 x ，它的概率密度函数为 $p(x)$ ，若积分 $H(x)=-\int p(x) \ln p(x)$ ，有意义，称它为连续型随机变量的熵。

从定义可以看出，两个随机变量具有相同分布时，它们的熵就相等，可见熵只与分布有关。

最大熵原则：无信息先验分布应取参数 θ 的变化范围内熵最大的分布。

可以证明，随机变量（或随机向量）的熵为最大的充分必要条件是随机变量（或随机向量）为均匀分布。因此，贝叶斯假设取无信息先验分布是“均匀分布”是符合信息论的最大熵原则的，它使随机变量（或随机向量）的熵为最大。现就随机变量取有限个值的情况加以证明。

定理 6.5 设随机变量 x 只取有限个值 a_1, a_2, \cdots, a_n ，相应的概率记为 p_1, p_2, \cdots, p_n ，则 x 的熵 $H(x)$ 最大的充分必要条件是： $p_1=p_2=\cdots=p_n=\frac{1}{n}$ 。

证明 考虑 $G(p_1, p_2, \cdots, p_n) = -\sum_{i=1}^n p_i \ln p_i + \lambda (\sum_{i=1}^n p_i - 1)$ ，为求其最大值，将 G 对 p_i 求偏微

商，并令偏微商为 0，得方程组：

$$0 = \frac{\partial G}{\partial p_i} = -\ln p_i - 1 + \lambda \quad (i=1, 2, \cdots, n)$$

求得 $p_1=p_2=\cdots=p_n$ 。又因为 $\sum_{i=1}^n p_i = 1$ ，所以 $p_1=p_2=\cdots=p_n=\frac{1}{n}$ 。此时相应的熵是

$$-\sum_{i=1}^n \frac{1}{n} \ln \frac{1}{n} = \ln n。反之，当 $p_1=p_2=\cdots=p_n$ 时， $G(p_1, p_2, \cdots, p_n)$ 取得最大值。$$

对于连续的随机变量也有同样的结果。

由此可见，在没有任何信息确定先验分布时，采用无信息先验分布是合理的。这时当然需要有较多的样本数据，进行较多次的计算才能得较好的结果。不过，无法指派先验概率分布的问题还是相当多，所以关于无信息先验分布的贝叶斯假设仍然有重要的意义。

3. 杰弗莱原则

杰弗莱对于先验分布的选取做出了重大的贡献，它提出一个不变原理，较好地解决了贝叶斯假设中的一个矛盾，并且给出了一个寻求先验密度的方法。杰弗莱原则由两个部分组成：一是对先验分布有一合理要求；一是给出具体的方法求得适合于要求的先验分布。

在贝叶斯假设中存在这样一个矛盾：如果对参数 θ 选用均匀分布，则对它的函数 $g(\theta)$ 作为参数时，也应选用均匀分布。反之当对 $g(\theta)$ 选用均匀分布时，参数 θ 也应选用均匀分布。然而上述的前提中，往往推导不出相应的结论。杰弗莱为了克服这一矛盾，提出了不变性要求，他认为一个合理的决定先验分布的准则应具有不变性。

若决定 θ 的先验分布为 $\pi(\theta)$ ，根据不变性准则，决定它的函数 $g(\theta)$ 的分布 $\pi_g(g(\theta))$ 应满足下面的关系：

$$\pi(\theta) = \pi_g(g(\theta)) |g'(\theta)| \quad (6.7)$$

问题的关键在于如何找到满足上面要求的先验分布 $\pi(\theta)$ 。杰弗莱巧妙地利用费歇信息阵的一个不变性质找到了要求的 $\pi(\theta)$ 。

参数 θ 的先验分布应以信息阵 $I(\theta)$ 行列式的平方根为核，即 $\pi(\theta) \propto |I(\theta)|^{1/2}$ ，其中

$$I(\theta) = E\left(\frac{\partial \ln p(x_1, x_2, \dots, x_n; \theta)}{\partial \theta}\right)\left(\frac{\partial \ln p(x_1, x_2, \dots, x_n; \theta)}{\partial \theta}\right)'$$

具体推导过程，这里不再给出，可参考相关文献。值得指出的是，杰弗莱原则是一个原则性的意见，用信息阵 $I(\theta)$ 行列式的平方根选取先验分布是一个具体的方法，这两者是不等同的，还可以寻求更适合体现这一准则的具体方法。

6.3.2 计算学习机制

任何系统经过运行能改善其行为，都是学习。到底贝叶斯公式求得的后验是否比原来信息有所改善呢？其学习的机制是什么？现以正态分布为例进行分析，从参数的变化看先验信息和样本数据在学习中所起的作用。

设 X_1, X_2, \dots, X_n 是来自正态分布 $N(\theta, \sigma_1^2)$ 的一个样本，其中 σ_1^2 已知， θ 未知。为了求 θ 的估计量 $\tilde{\theta}$ ，取另一个正态分布 $N(\mu_0, \sigma_0^2)$ 作为该正态均值 θ 的先验分布，即取先验为： $\pi(\theta) = N(\mu_0, \sigma_0^2)$ 。用贝叶斯公式可以计算出后验仍为正态分布： $h(\theta | \bar{x}_1) = N(\alpha_1, d_1^2)$ ，其中：

$$\bar{x}_1 = \sum_{i=1}^n \frac{x_i}{n}, \quad \alpha_1 = \left(\frac{1}{\sigma_0^2} \mu_0 + \frac{n}{\sigma_1^2} \bar{x}_1\right) / \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma_1^2}\right), \quad d_1^2 = \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma_1^2}\right)^{-1}.$$

用后验 $h(\theta | \bar{x}_1)$ 的数学期望 α_1 作为 θ 的估计值，有：

$$\tilde{\theta} = E(\theta | \bar{x}_1) = \left(\frac{1}{\sigma_0^2} \mu_0 + \frac{n}{\sigma_1^2} \bar{x}_1\right) \cdot d_1^2 \quad (6.8)$$

由此可见，这样得到的 θ 的估计值 $\tilde{\theta}$ 是先验分布中的期望 μ_0 与样本均值 \bar{x}_1 的加权平均。因为 σ_0^2 是 $N(\mu_0, \sigma_0^2)$ 的方差，它的倒数 $1/\sigma_0^2$ 就是 μ_0 的精度。样本均值 \bar{x}_1 的方差是 σ_1^2/n ，它的倒数 n/σ_1^2 就是样本均值 \bar{x} 的精度。可知 $\tilde{\theta}$ 是将 μ_0 与 \bar{x}_1 按各自的精度加权平均。方差越小者精度越高，在后验均值中所占的比重越大。此外，样本的容量 n 越大则 σ_0^2/n 越小，则样本均值 \bar{x}_1 在后验均值中所占的比重越大。当 n 相当大时，先验均值在后验中的影响将变得很小。这说明贝叶斯公式求出的后验确实对先验信息和样本数据进行了合理的综合，其得到的结果比单独使用先验信息或样本数据都更完善，其学习机制确实是有效的。在采用其它共轭先验分布的情况下，也有类似的结果。

从前面的讨论可知，在共轭先验的前提下，可以将得到的后验信息作为新一轮计算的先验，与进一步获得的样本信息综合，求得下一个后验信息。如果多次重复这个过程，得到的后验信息是否越来越接近于实际结果？对这个问题可作如下分析：

用计算得到的后验分布 $h(\theta | \bar{x}_1) = N(\alpha_1, d_1^2)$ 作为新一轮计算的先验时，设新的样本 X_1, X_2, \dots, X_n 来自正态分布 $N(\theta, \sigma_2^2)$ ，其中 σ_2^2 已知， θ 待估计。则新的后验分布为： $h_1(\theta | \bar{x}_2) = N(\alpha_2, d_2^2)$ ，

$$\text{其中：} \bar{x}_2 = \sum_{i=1}^n \frac{x_i}{n}, \quad \alpha_2 = \left(\frac{1}{d_1^2} \alpha_1 + \frac{n}{\sigma_2^2} \bar{x}_2 \right) / \left(\frac{1}{d_1^2} + \frac{n}{\sigma_2^2} \right), \quad d_2^2 = \left(\frac{1}{d_1^2} + \frac{n}{\sigma_2^2} \right)^{-1}.$$

用后验 $h_1(\theta | \bar{x}_2)$ 的数学期望 $\alpha_2 = \left(\frac{1}{\sigma_0^2} \mu_0 + \frac{n}{\sigma_1^2} \bar{x} \right) / \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma_1^2} \right)$ 作为 θ 的估计值，由于

$$\alpha_1 = \left(\frac{1}{\sigma_0^2} \mu_0 + \frac{n}{\sigma_1^2} \bar{x}_1 \right) \cdot d_1^2, \text{ 计算可得:}$$

$$\begin{aligned} \alpha_2 &= \left(\frac{1}{d_1^2} \alpha_1 + \frac{n}{\sigma_2^2} \bar{x}_2 \right) \cdot d_2^2 = \left(\frac{1}{\sigma_0^2} \mu_0 + \frac{n}{\sigma_1^2} \bar{x}_1 + \frac{n}{\sigma_2^2} \bar{x}_2 \right) \cdot d_2^2 \\ &= \left(\frac{1}{\sigma_0^2} \mu_0 + \frac{n}{\sigma_1^2} \bar{x}_1 \right) \cdot d_2^2 + \frac{n}{\sigma_2^2} \bar{x}_2 \cdot d_2^2 \end{aligned} \quad (6.9)$$

$$\text{又由于 } \frac{n}{\sigma_2^2} > 0, \text{ 故 } d_2^2 = \left(\frac{1}{d_1^2} + \frac{n}{\sigma_2^2} \right)^{-1} = \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma_1^2} + \frac{n}{\sigma_2^2} \right)^{-1} < d_1^2 = \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma_1^2} \right)^{-1}$$

可知在 α_2 中， $\left(\frac{1}{\sigma_0^2} \mu_0 + \frac{n}{\sigma_1^2} \bar{x}_1 \right) \cdot d_2^2 < \alpha_1$ ，也就是说，由于新样本的加入，先验和旧样本所

占的比重降低。由式 (6.9) 容易看出，当新的样本（不失一般性，假定容量相同）继续增加，将有：

$$\alpha_m = \left(\frac{1}{\sigma_0^2} \mu_0 + \frac{n}{\sigma_1^2} \bar{x}_1 + \frac{n}{\sigma_2^2} \bar{x}_2 + \dots + \frac{n}{\sigma_m^2} \bar{x}_m \right) \cdot d_m^2$$

$$= \left(\frac{1}{\sigma_0^2} \mu_0 + \sum_{k=1}^m \frac{n}{\sigma_k^2} \bar{x}_k \right) \cdot d_m^2 \quad (k=1, 2, \dots, m) \quad (6.10)$$

由(6.10)式可知,如果所有新的样本的方差相同,则等同于一个容量为 $m \times n$ 的样本。以上过程将先验和各样本均值按各自的精度加权平均,精度越高者其权值越大。由此可见,如果能正确估计先验分布密度,就可以使用少量样本数据,进行少量计算而得到较满意的结果。这在样本难得的情形特别有用,这也是贝叶斯学习优于其它方法之处。因此先验分布的指派在贝叶斯学习中有重要的意义。如果没有任何先验信息而采用无信息先验分布时,随着使用的样本增多,样本信息的影响越来越显著。在样本的噪音很小的前提下,得到的后验信息也将越来越接近于实际,只不过需要大量的计算而已。

6.3.3 贝叶斯问题求解步骤

贝叶斯问题求解的基本步骤可以概括为:

(1) 定义随机变量。将未知参数看成随机变量(或随机向量),记为 θ 。将样本 X_1, X_2, \dots, X_n 的联合分布密度 $p(X_1, X_2, \dots, X_n; \theta)$ 看成是 X_1, X_2, \dots, X_n 对 θ 的条件分布密度,记为 $p(X_1, X_2, \dots, X_n | \theta)$ 或 $p(D | \theta)$ 。

(2) 确定先验分布密度 $p(\theta)$ 。采用共轭先验分布。如果对先验分布没有任何信息,就采用无信息先验分布的贝叶斯假设。

(3) 利用贝叶斯定理计算后验分布密度。

(4) 利用计算得到的后验分布密度对所求问题作出推断。

以单变量单个参数情形为例,考虑“抛掷图钉问题”:将图钉抛到空中,图钉落下静止后将取以下两种状态之一:头(head)着地或尾(tail)着地。假设我们抛图钉 $N+1$ 次,问从前 N 次的结果如何决定第 $N+1$ 次出现头的概率。

第一步,定义随机变量 θ ,其值 θ 对应于抛图钉头着地的物理概率可能的真值。密度函数 $p(\theta)$ 表示我们对 θ 的不确定性。第 I 次抛掷结果的变量为 X_I ($I=1, 2, \dots, N+1$),观测值的集合为 $D=\{X_1=x_1, \dots, X_n=x_n\}$ 。于是将问题表示为由 $p(\theta)$ 计算 $p(x_{N+1}|D)$ 。

第二步,用贝叶斯定理获得给定 D 时 θ 的概率分布: $p(\theta|D) = \frac{p(\theta)p(D|\theta)}{p(D)}$, 其中

$p(D) = \int p(D|\theta)p(\theta)d\theta$, $p(D|\theta)$ 是二项分布样本的似然函数。如果已知 θ 的值(即参数 θ),则 D 中的观测值是相互独立的,并且任何一次观测出现头的概率是 θ ,出现尾的概率为 $(1-\theta)$ 。于是:

$$p(\theta|D) = \frac{p(\theta)\theta^h(1-\theta)^t}{p(D)} \quad (6.11)$$

其中 h 和 t 分别是在 D 中观测到的头和尾的次数,称为二项分布样本的充分统计量。

第三步,求所有 θ 的所有可能的值的平均值,作为第 $N+1$ 次抛掷图钉出现头的概率:

$$\begin{aligned} p(X_{N+1} = heads | D) &= \int p(X_{N+1} = heads | \theta)p(\theta | D)d\theta \\ &= \int \theta \cdot p(\theta | D)d\theta \equiv E_{p(\theta|D)}(\theta) \end{aligned} \quad (6.12)$$

其中 $E_{p(\theta|D)}(\theta)$ 表示 θ 对于分布 $p(\theta|D)$ 的数学期望。

第四步，为 θ 指派先验分布和超参数。

指派先验通常采用的方法是先假定先验的分布，再确定分布的参数。假定先验是 Beta 分布：

$$p(\theta) = \text{Beta}(\theta | \alpha_h, \alpha_t) = \frac{\Gamma(\alpha)}{\Gamma(\alpha_h)\Gamma(\alpha_t)} \theta^{\alpha_h-1} (1-\theta)^{\alpha_t-1} \quad (6.13)$$

其中 $\alpha_h > 0$ 和 $\alpha_t > 0$ 是 Beta 分布的参数， $\alpha = \alpha_h + \alpha_t$ ， $\Gamma(\cdot)$ 是 Gamma 函数。为了和参数 θ 相区别，将 α_h 和 α_t 称为“超参数”。因为 Beta 分布属于共轭分布族，得到的后验也是 Beta 分布：

$$p(\theta|D) = \frac{\Gamma(\alpha + N)}{\Gamma(\alpha_h + h)\Gamma(\alpha_t + t)} \theta^{\alpha_h+h-1} (1-\theta)^{\alpha_t+t-1} = \text{Beta}(\theta | \alpha_h + h, \alpha_t + t) \quad (6.14)$$

对于这个分布， θ 的数学期望有一个简单的形式：

$$\int \theta \cdot \text{Beta}(\theta | \alpha_h, \alpha_t) d\theta = \frac{\alpha_h}{\alpha} \quad (6.15)$$

于是，给定一个 Beta 先验，我们得到第 $N+1$ 次抛掷出现头的概率的一个简单的表达式：

$$p(X_{N+1} = \text{heads} | D) = \frac{\alpha_h + h}{\alpha + N} \quad (6.16)$$

确定先验 $p(\theta)$ 的 Beta 分布的超参数有多种方法。例如“想象将来数据法”和“等价样本法”。其他方法见 Winkler、Chaloner 和 Duncan 的论文。使用想象将来数据法，由 (6.16) 式可得到两个方程，便可求出 Beta 分布的两个参数（超参数） α_h 和 α_t 。

在单变量多参数（一种变量有多种状态）的情形，一般设 X 是带有高斯分布的连续变量，假设有物理概率分布 $p(x|\theta)$ ，则有：

$$p(x|\theta) = (2\pi\nu)^{-1/2} e^{-(x-\mu)^2/2\nu} \quad (\text{其中 } \theta = \{\mu, \nu\})$$

依照二项分布的做法，赋予参数变量以先验，应用贝叶斯定理由给定观测样本的数据集 $D = \{X_1=x_1, X_2=x_2, \dots, X_N=x_N\}$ 求出后验，也就是学习这些参数：

$$p(\theta|D) = p(D|\theta) p(\theta) / p(D)$$

然后取 θ 的可能的值的平均值作预测：

$$p(x_{N+1}|D) = \int p(x_{N+1}|\theta) p(\theta|D) d\theta \quad (6.17)$$

对指数家族而言，计算是特别有效的而且是封闭的。在多项样本情形，如果 X 的观察值是离散的，使用 Dirichlet 分布作为共轭先验可以使计算简化。

贝叶斯定理的计算学习机制是将先验分布中的期望值与样本均值按各自的精度进行加权平均，精度越高者其权值越大。在先验分布为共轭分布的前提下，可以将后验信息作为新一轮计算的先验，用贝叶斯定理与进一步得到的样本信息进行综合。多次重复这个过程后，样本信息的影响越来越显著。由于贝叶斯方法可以综合先验信息和后验信息，既可避免只使用先验信息可能带来的主观偏见，和缺乏样本信息时的大量盲目搜索与计算，也可避免只使用后验信息带来的噪音的影响。因此，适用于具有概率统计特征的数据采掘和知识发现问题，尤其是样本难以取得或代价昂贵的问题。合理准确地确定先验，是贝叶斯方法进行有效学习的关键问题。目前先验分布的确定依据只是一些准则，没有可操作的完整的理论，在许多情况下先验分布的合理性和准确性难以评价。对于这些问题还需要进一步深入研究。

6.4 简单贝叶斯学习模型

简单贝叶斯学习模型 (Simple Bayes 或 Naïve Bayes) 将训练实例 I 分解成特征向量 X 和决策类别变量 C 。简单贝叶斯模型假定特征向量的各分量间相对于决策变量是相对独立的,也就是说各分量独立地作用于决策变量。尽管这一假定一定程度上限制了简单贝叶斯模型的适用范围,然而在实际应用中,不仅以指数级降低了贝叶斯网络构建的复杂性,而且在许多领域,在违背这种假定的条件下,简单贝叶斯也表现出相当的健壮性和高效性[Nigam 98],它已经成功地应用到分类、聚类及模型选择等数据挖掘的任务中。目前,许多研究人员正致力于改善特征变量间独立性的限制[Heckerman 97],以使它适用于更大的范围。

6.4.1 简单贝叶斯学习模型

贝叶斯定理告诉我们如何通过给定的训练样本集预测未知样本的类别,它的预测依据就是取后验概率

$$P(C_i | A) = \frac{P(C_i) * P(A | C_i)}{P(A)} \quad (6.18)$$

最大的类别。这里 E 是测试样本, $P(Y | X)$ 是在给定 X 的情况下 Y 的条件概率。等式右侧的概率都是从样本数据中估计得到的。设样本表示成属性向量,如果属性对于给定的类别独立,那么 $P(A | C_i)$ 可以分解成几个分量的积: $P(a_1 | C_i) * P(a_2 | C_i) * \dots * P(a_m | C_i)$ (这里 v_i 是样本 E 的第 i 个属性)。从而后验概率的计算公式为:

$$P(C_i | A) = \frac{P(C_i)}{P(A)} \prod_{j=1}^m P(a_j | C_i) \quad (6.19)$$

这个过程称之为简单贝叶斯分类 (SBC: Simple Bayesian Classifier)。一般认为,只有在独立性假定成立的时候, SBC 才能获得精度最优的分类效率;或者在属性相关性较小的情况下,能获得近似最优的分类效果。然而这种较强的限制条件,似乎与 SBC 在许多领域惊人的性能很不一致,其中包括属性具有明显依赖性的情况。在 UCI 的 28 个数据集中,其中 16 个要好于 C4.5 的性能,与 CN2 和 PEBLS 基本相似,许多研究也得到了类似的结论[Clark & Niblett, 1989; Dougherty Kohavi & Sahami, 1995]。同时也有一些研究人员提出了一些策略以改善属性之间独立性的限制[Nigam 98],获得了一定的成功。

在 (6.19) 中的概率可采用样本的最大似然估计:

$$P(v_j | C_i) = \frac{\text{count}(v_j \wedge c_i)}{\text{count}(c_i)} \quad (6.20)$$

为了防止 (6.19) 出现 0 的情况,当实际计算为零时,可以直接指定 (6.19) 的结果为 $0.5/N$, 这里 N 为例子总数。

假定只有两类,将这两类分别称为 0 类和 1 类: a_1 到 a_k 表示一个给定测试集的属性;

$$b_0 = P(C = 0), b_1 = P(C = 1) = 1 - b_0, p_{j0} = P(A_j = a_j | C = 0), p_{j1} = P(A_j = a_j | C = 1)$$

则：

$$p = P(C = 1 | A_1 = a_1 \wedge \cdots \wedge A_k = a_k) = \left(\prod_{j=1}^k p_{j1} \right) b_1 / z \quad (6.21)$$

$$q = P(C = 0 | A_1 = a_1 \wedge \cdots \wedge A_k = a_k) = \left(\prod_{j=1}^k p_{j0} \right) b_0 / z \quad (6.22)$$

这里 z 是一个固定常量。将上面两式两边取对数后相减得：

$$\log p - \log q = \left(\sum_{j=1}^k \log p_{j1} - \log p_{j0} \right) + \log b_1 - \log b_0 \quad (6.23)$$

这时，取 $w_j = \log p_{j1} - \log p_{j0}$, $b = \log b_1 - \log b_0$ ，则上式转化为：

$$\log (1-p) / p = - \sum_{j=1}^k w_j - b \quad (6.24)$$

两边取指数，并且重新排列：

$$p = \frac{1}{1 + e^{-\sum_{j=1}^k w_j - b}} \quad (6.25)$$

为计算该式，一般来说，假定属性 A_j 有 $v(j)$ 个可能的属性值，那么当：

$$w_{jj'} = \log P(A_j = a_{jj'} | c = 1) - \log P(A_j = a_{jj'} | c = 0) \quad (1 \leq j' \leq v(j)) \quad (6.26)$$

时，上式转化为：

$$P(C(x) = 1) = \frac{1}{1 + e^{-\left(\sum_{j=1}^k \sum_{j'=1}^{v(j)} I(A_j(x) = a_{jj'}) w_{jj'} - b \right)}} \quad (6.27)$$

这里， I 是一个示性函数：如果 φ 为真，那么 $I(\varphi) = 1$ ，否则： $I(\varphi) = 0$ 。

在实际计算中，上式只需根据公式 (6.20) 的思想即可计算出结果。

公式 (6.27) 实际上是一个具有 sigmoid 型激励的感知器函数。该函数分别输入每一个属性 A_j 的 $v(j)$ 个可能的值。因此简单贝叶斯分类器在特定输入场合下的表达能力与感知器模型是等价的。进一步的研究表明，简单贝叶斯分类器可以推广到针对数值属性的逻辑回归方法。

在此考虑公式 (6.20)，如果 A_j 具有离散值，那么 $\text{count}(A_j = a_j \wedge C = c_i)$ 能够从训练例子中直接计算出来。如果 A_j 是连续值，那么就需要离散化该属性，使用非监督离散化将该属性离散化到 M 个等宽度的区间内，一般 $M = 10$ 。也可以使用更复杂的离散化方法，如监督离散化方法。

让每一个 A_j 是一个数值属性（离散或连续），逻辑回归假设模型：

$$\log \frac{P(C=1 | A_1=a_1, \dots, A_k=a_k)}{P(C=0 | A_1=a_1, \dots, A_k=a_k)} = \sum_{j=1}^k b_j a_j + b_0 \quad (6.28)$$

对上式采用如 (6.24) 式同样的变换后, 得到:

$$p = \frac{1}{1 + e^{-\sum_{j=1}^k b_j a_j - b_0}}, \quad (6.29)$$

显然, 这也是具有 sigmoid 型激励的感知器函数, 每个属性作为输入的单节点的感知器, 并且每个属性值以它们的大小编码。用一个 $\varphi = a_j$ 的函数 $f(\varphi)$ 替换 $b_j a_j$, 如果 A_j 的范围被分割为 M 个部分, 那么第 i 部分在 $[c_{j(i-1)}, c_{ji}]$, 则该函数就是:

$$b_j a_j = f_j(a_j) = \sum_{i=1}^M b_{ji} I(c_{j(i-1)} \leq c_{ji}) \quad (6.30)$$

这里每一个 b_{ji} 是一个常数, 综合公式 (6.29) (6.30) 可以得到:

$$P(C(x)=1) = \frac{1}{1 + e^{-(\sum_{j=1}^k \sum_{i=1}^M b_{ji} I(c_{j(i-1)} \leq c_{ji})) - b_0}} \quad (6.31)$$

这就是最终的回归函数。因此简单贝叶斯分类是逻辑回归的一种非参数化、非线性的扩展。

通过设置 $b_{ji} = (c_{j(i-1)} + c_{ji}) / (2b_j)$, 能够近似得到标准逻辑回归公式。

6.4.2 简单贝叶斯模型的提升

提升方法 (Boosting) 总的思想是学习一系列分类器, 在这个序列中每一个分类器对它前一个分类器导致的错误分类例子给与更大的重视。尤其是, 在学习完分类器 H_k 之后, 增加了由 H_k 导致分类错误的训练例子的权值, 并且通过重新对训练例子计算权值, 再学习下一个分类器 H_{k+1} 。这个过程重复 T 次。最终的分类器从这一系列的分类器中综合得出。

在这个过程中, 每个训练例子被赋予一个相应的权值, 如果一个训练例子被分类器错误分类, 那么就相应地增加该粒子的权重, 使得在下一次学习中, 分类器对该例代表的情况更加重视。

提升算法对两类分类问题的处理是通过 Freund 和 Scapire 给出的 AdaBoost 算法实现的 [Freund 95]。

算法 6.1 AdaBoost 算法。

Input:

N 个训练实例: $\langle (x_1, y_1), \dots, (x_N, y_N) \rangle$

N 个训练实例上的分布 $D: w$, w 为训练实例的权向量。

T 为训练重复的趟数。

1. *Initialize*:
2. 初始化训练实例的权向量。 $w_i = 1/N \quad i = (1 \dots N)$
3. *for* $t = 1$ *to* T
4. 给定权值 w_i^t 得到一个假设 $H^{(t)} : X \rightarrow [0,1]$
5. 估计假设 $H^{(t)}$ 的总体误差, $e^{(t)} = \sum_{i=1}^N w_i^{(t)} |y_i - h_i^{(t)}(x_i)|$
6. 计算 $\beta^{(t)} = e^{(t)} / (1 - e^{(t)})$
7. 计算下一轮样本的权值 $w_i^{(t+1)} = w_i^{(t)} (\beta^{(t)})^{1 - |y_i - h_i^{(t)}(x_i)|}$
8. 正规化 $w_i^{(t+1)}$, 使其总和为 1
9. *End for*
10. *Output* :
11.
$$h(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^T (\log \frac{1}{\beta^{(t)}}) h^{(t)}(x) \geq \frac{1}{2} \sum_{t=1}^T (\log \frac{1}{\beta^{(t)}}) \\ 0 & \text{otherwise} \end{cases}$$

在这里假设每一个分类器都是实际有用的, $e^t < 0.5$, 也就是说, 在每一次分类的结果中, 正确分类的样本个数始终大于错误分类的样本个数。可以看出, $\beta^{(t)} < 1$, 当 $|y_i - h_i^{(t)}(x_i)|$ 增加时, $w_i^{(t+1)}$ 也增加, 因此算法满足了提升的思想。

对算法的几点说明:

- (1) $h^{(t)}(x)$ 采用公式 (6.27) 计算得出, 计算的结果或者为 0 或者为 1。
- (2) 公式 (6.20) 中, 关于条件概率的计算 $P(A_j = a_{jj} | C = c)$ 。对 (6.20) 中, 在未考虑权值的情况下, $count(condition)$ 计算操作的基准是 1, 例如有 k 个例子满足 $condition$ 那么 $count(condition) = k$ 。在考虑权值的情况下, $count(condition)$ 计算操作的基准变为每个例子的权值, 如果有 k 个例子满足 $condition$, 那么 $count(condition) = \sum_i^k w_i$ 。通过调整权值, 体现了提升的思想。

- (3) 所谓算法的输出是指：当给定一个新的 x ，通过算法的第(6)步，就可以利用以前每次增强学习的结果，通过投票输出最终的结果。

算法最终的联合假设可以定义为：

$$H(x) = \frac{1}{1 + \prod_{t=1}^T (\beta^{(t)})^{2r(x)-1}} \quad \text{其中 } r(x) = \frac{\sum_{t=1}^T (\log 1/\beta^t) H^{(t)}(x)}{\sum_{t=1}^T (\log 1/\beta^t)}$$

现在可以说明，提升后的简单贝叶斯分类器在表达能力上相当于具有单个隐含层的多层感知器模型。令 $\alpha = \prod_{t=1}^T \beta^t$ ， $v^{(t)} = \log \beta^{(t)} / \log \alpha$ ，那么

$$H(x) = \frac{1}{1 + \alpha^{2(\sum_{t=1}^T v^{(t)} H^{(t)}(x)) - 1}} = \frac{1}{1 + e^{\sum_{t=1}^T 2 \log \beta^{(t)} H^{(t)}(x) - \sum_{t=1}^T \log \beta^{(t)}}}$$

即，单个分类器的输出乘以权值的和应用到一个 *sigmoid* 函数构成联合分类器的输出。

既然单个简单贝叶斯分类器的输出等价于一个感知器，那么联合分类器等价于一个具有单隐含层的感知器网络。

对多类分类问题的提升贝叶斯方法如下：

算法 6.2 多类分类 AdaBoost 算法。

Input:

N 个训练实例： $\langle (x_1, y_1), \dots, (x_N, y_N) \rangle$

N 个训练实例上的分布 $D: w$ ， w 为训练实例的权向量。

T 为训练重复的趟数。

1. *Initialize:*

2. 初始化训练实例的权向量。 $w_i = 1/N \quad i = (1 \dots N)$

3. *for* $t = 1$ *to* T

4. 给定权值 w_i^t 得到一个假设 $H^{(t)}: X \rightarrow Y$

5. 估计假设 $H^{(t)}$ 的总体误差， $e^{(t)} = \sum_{i=1}^N w_i^{(t)} I(y_i \neq h_i^{(t)}(x_i))$

6. 计算 $\beta^{(t)} = e^{(t)} / (1 - e^{(t)})$

7. 计算下一轮样本的权值 $w_i^{(t+1)} = w_i^{(t)} (\beta^{(t)})^{1 - I(y_i = h_i^{(t)}(x_i))}$

8. 正规化 $w_i^{(t+1)}$ ，使其总和为 1

9. *End for*

10. *Output:*

11. $h(x) = \arg \max_{y \in Y} \sum_{t=1}^T (\log \frac{i}{\beta^{(t)}}) I(h^{(t)}(x) = y)$

这里 $I(\phi) = 1$ ，如果 $\phi = T$ ；否则 $I(\phi) = 0$

6.4.2 提升简单贝叶斯分类的计算复杂性

假设样本空间中个体样本具有 f 个属性，每个属性有 v 个属性值。那么有公式(6.27)导出的一个简单贝叶斯分类器有 $fv+1$ 个参数，这些参数被累计学习 $2fv+2$ 次，在每次学习中，每个训练例子的每个属性值都会使得精度得以提高。因此， n 个训练实例的时间复杂度就是 $O(nf)$ ，与 v 无关。这样的时间复杂度本质上就是最佳的。提升后的简单贝叶斯，每一趟的时间复杂度也是 $O(nf)$ 。对 T 趟训练，其复杂度为 $O(Tnf)$ ，而 T 是一常数，所以它的时间复杂度仍然为 $O(nf)$ 。

对简单贝叶斯分类器，其主要计算是计数计算。训练例子可以从磁盘或磁带机中顺序地进行处理，也可以分批地处理。因此该方法非常适合在大数据量的知识发现。数据集可以不需全部装入内存，仍旧可以保存在磁盘或磁带上。然而这种提升方法应用到简单贝叶斯模型中存在如下问题：

1. 从提升的思想来看，当训练集中存在噪音时，提升的方法会把这些噪音当作有用的信息通过权值而放大，这会降低提升的性能。当噪音很多时，这种提升会导致更糟的结果。
2. 虽然理论上提升方法可以保证训练集的差错率为 0，但当其运用到简单贝叶斯模型时，却不能保证是这样，训练集的差错始终存在。

6.5 贝叶斯网络的建造

6.5.1 贝叶斯网络的结构及建立方法

简而言之，贝叶斯网络是一个带有概率注释的有向无环图。这个图模型能表示大的变量集合的联合概率分布（物理的或贝叶斯的），可以分析大量变量之间的相互关系，利用贝叶斯定理揭示的学习和统计推断功能，实现预测、分类、聚类、因果分析等数据挖掘任务。

关于一组变量 $\mathbf{X}=\{X_1, X_2, \dots, X_n\}$ 的贝叶斯网络由以下两部分组成：（1）一个表示 \mathbf{X} 中的变量的条件独立断言的网络结构 S ；（2）与每一个变量相联系的局部概率分布集合 P 。两者定义了 \mathbf{X} 的联合概率分布。 S 是一个有向无环图， S 中的节点一对一地对应于 \mathbf{X} 中的变量。以 X_i 表示变量节点， Pa_i 表示 S 中的 X_i 的父节点。 S 的节点之间缺省弧线则表示条件独立。 \mathbf{X} 的联合概率分布表示为：

$$p(\mathbf{x}) = \prod_{i=1}^n p(x_i | pa_i) \quad (6.32)$$

以 P 表示 (6.31) 式中的局部概率分布, 即乘积中的项 $p(x_i | \mathbf{Pa}_i)$ ($i=1, 2, \dots, n$), 则二元组 (S, P) 表示了联合概率分布 $p(\mathbf{X})$ 。当仅仅从先验信息出发建立贝叶斯网络时, 该概率分布是贝叶斯的 (主观的)。当从数据出发进行学习, 进而建立贝叶斯网络时, 该概率是物理的 (客观的)。

为了建立贝叶斯网络, 第一步, 必须确定为建立模型有关的变量及其解释。为此, 需要: (1) 确定模型的目标, 即确定问题相关的解释; (2) 确定与问题有关的许多可能的观测值, 并确定其中值得建立模型的子集; (3) 将这些观测值组织成互不相容的而且穷尽所有状态的变量。这样做的结果不是唯一的。

第二步, 建立一个表示条件独立断言的有向无环图。根据概率乘法公式有:

$$\begin{aligned} p(\mathbf{x}) &= \prod_{i=1}^n p(x_i | x_1, x_2, \dots, x_{i-1}) \\ &= p(x_1)p(x_2 | x_1)p(x_3 | x_1, x_2) \dots p(x_n | x_1, x_2, \dots, x_{n-1}) \end{aligned} \quad (6.33)$$

对于每个变量 X_i , 如果有某个子集 $\Pi i \subseteq \{X_1, X_2, \dots, X_{i-1}\}$ 使得 X_i 与 $\{X_1, X_2, \dots, X_{i-1}\} \setminus \Pi i$ 是条件独立的, 即对任何 \mathbf{X} , 有:

$$p(x_i | x_1, x_2, \dots, x_{i-1}) = p(x_i | \pi_i) \quad (i=1, 2, \dots, n) \quad (6.34)$$

则由 (6.33) (6.34) 两式可得: $p(\mathbf{x}) = \prod_{i=1}^n p(x_i | \pi_i)$ 。变量集合 $(\Pi 1, \dots, \Pi n)$ 对应于父节点 $(\mathbf{Pa}_1, \dots, \mathbf{Pa}_n)$, 故又可以写成: $p(\mathbf{x}) = \prod_{i=1}^n p(x_i | \mathbf{pa}_i)$ 。于是, 为了决定贝叶斯网络的结构, 需要 (1) 将变量 X_1, X_2, \dots, X_i 按某种次序排序; (2) 决定满足 (6.34) 式的变量集 Πi ($i=1, 2, \dots, n$)。

从原理上说, 如何从 n 个变量中找出适合条件独立的顺序, 是一个组合爆炸问题。因为要比较 $n!$ 种变量顺序。不过, 通常可以在现实问题中决定因果关系, 而且因果关系一般都对应于条件独立的断言。因此, 可以从原因变量到结果变量划一个带箭头的弧来直观表示变量之间的因果关系。

第三步, 指派局部概率分布 $p(x_i | \mathbf{Pa}_i)$ 。在离散的情形, 需要为每一个变量 X_i 的各个父节点的状态指派一个分布。

显然, 以上各步可能交叉进行, 而不是简单的顺序进行可以完成的。

6.5.2 学习贝叶斯网络的概率分布

现在考虑这样的问题: 给定贝叶斯网络的结构, 如何利用给定样本数据去学习网络的概率分布, 即更新网络变量原有的先验分布。这里使用的是贝叶斯方法, 既综合先验知识和数据去改进已有知识的技术, 这些技术可用于数据采掘。假设变量组 $\mathbf{X} = (X_1, X_2, \dots, X_n)$ 的物理联合概率分布可以编码在某个网络结构 S 中:

$$p(\mathbf{x} | \theta_S, S^d) = \prod_{i=1}^n p(x_i | \mathbf{pa}_i, \theta_i, S^d) \quad (6.35)$$

其中 θ_i 是分布 $p(x_i | \mathbf{pa}_i, \theta_i, S^d)$ 的参数向量, θ_S 是参数组 $(\theta_1, \theta_2, \dots, \theta_n)$ 的向

量, 而 S^b 表示物理联合分布可以依照 S 被分解的假设。需要说明的是这个分解是不交叉 (重叠) 的。例如, 给定 $\mathbf{X}=\{X_1, X_2\}$, \mathbf{X} 的任何联合分布可以被分解成对应于没有弧的网络结构, 也可以被分解成对应于 $X_1 \rightarrow X_2$ 的网络结构。这就是交叉 (重叠)。此外, 假设从 \mathbf{X} 的物理联合概率分布得到一个随机样本 $D=\{X_1, \dots, X_n\}$ 。 D 的一个元素 X_i 表示样本的一个观测值, 称为一个案例。定义一个取向量值的变量 θ_s 对应于参数向量 θ_s , 并指派一个先验概率密度函数 $p(\theta_s | S^b)$ 表示对 θ_s 的不确定性, 于是贝叶斯网络的学习概率问题可以简单地表示成: 给定随机样本 D , 计算后验分布 $p(\theta_s | D, S^b)$ 。

下面用无约束多项分布来讨论学习概率的基本思想。假定每个变量 $X \in \mathbf{X}_n$ 是离散的, 有 r_i 个可能的值 $x_i^1, x_i^2, \dots, x_i^{r_i}$, 每个局部分布函数是一组多项分布的集合, 一个分布对应于 \mathbf{pa}_i 的一个构成 (即一个分量)。也就是说, 假定

$$p(x_i^k | \mathbf{pa}_i^j, \theta_i, S^b) = \theta_{ijk} > 0 \quad (i=1, 2, \dots, n; j=1, 2, \dots, q_i; k=1, 2, \dots, r_i) \quad (6.36)$$

其中 $\mathbf{pa}_i^1, \mathbf{pa}_i^2, \dots, \mathbf{pa}_i^{q_i}$ 表示 \mathbf{pa}_i 的构成, $q_i = \prod_{X_i \in \mathbf{Pa}_i} r_i$ 。 $\theta_i = ((\theta_{ijk})_{k=2}^{r_i})_{j=1}^{q_i}$ 是参数, θ_{ij1} 没有列入,

因为 $\theta_{ij1} = 1 - \sum_{k=2}^{r_i} \theta_{ijk}$, 可以通过计算得到。为方便起见, 定义参数向量:

$$\theta_{ij} = (\theta_{ij2}, \theta_{ij3}, \dots, \theta_{ijr_i}) \quad (i=1, 2, \dots, n; j=1, 2, \dots, q_i)$$

给定以上的局部分布函数后, 需要有以下两个假设, 才可以以封闭的形式计算后验分布 $p(\theta_s | D, S^b)$:

(1) 在随机样本 D 中没有缺损数据, 这时又称 D 是完全的;

(2) 参数向量 θ_{ij} 是相互独立的, 即 $p(\theta_s | S^b) = \prod_{i=1}^n \prod_{j=1}^{q_i} p(\theta_{ij} | S^b)$ 。这就是参数独立假设。

在以上两个假设下, 对于给定的随机样本 D , 参数仍然保持独立:

$$p(\theta_s | D, S^b) = \prod_{i=1}^n \prod_{j=1}^{q_i} p(\theta_{ij} | D, S^b) \quad (6.37)$$

于是可以相互独立地更新每一个参数向量 θ_{ij} 。假设每一个参数向量 θ_{ij} 有先验 Dirichlet 分布 $\text{Dir}(\theta_{ij} | \alpha_{ij1}, \alpha_{ij2}, \dots, \alpha_{ijr_i})$, 得到后验分布为:

$$p(\theta_{ij} | D, S^b) = \text{Dir}(\theta_{ij} | \alpha_{ij1} + N_{ij1}, \alpha_{ij2} + N_{ij2}, \dots, \alpha_{ijr_i} + N_{ijr_i}) \quad (6.38)$$

其中 N_{ijk} 是当 $X_i = x_i^k$ 且 $\mathbf{pa}_i = \mathbf{pa}_i^j$ 时 D 中的案例数目。

现在可以求 θ_s 的可能的构成的平均值来得到感兴趣的预测。例如, 计算 D 中第 $N+1$ 个案例

例 $p(X_{N+1} | D, S^b) = \int \prod_{i=1}^n \theta_{ijk} p(\theta_s | D, S^b) d\theta = \int \prod_{i=1}^n \theta_{ijk} p(\theta_{ij} | D, S^b) d\theta_{ij}$ 。利用参数对给定 D 保持独立, 可以计算数学期望:

$$p(X_{N+1} | D, S^b) = \int \prod_{i=1}^n \theta_{ijk} p(\theta_s | D, S^b) d\theta = \prod_{i=1}^n \int \theta_{ijk} p(\theta_{ij} | D, S^b) d\theta_{ij}$$

通过计算, 最后可得: $p(X_{N+1} | D, S^b) = \prod_{i=1}^n \frac{\alpha_{ijk} + N_{ijk}}{\alpha_{ij} + N_{ij}}$ (6.39)

其中 $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$ 且 $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ 。由于无约束多项分布属于指数家族，上面的计算将变得简单。

关于变量组 \mathbf{X} 的贝叶斯网络表示了 \mathbf{X} 的联合概率分布，所以，无论是从先验知识、数据或两者的综合建立的贝叶斯网络，原则上都可以用它来推断任何感兴趣的概率。不过，离散变量的任意贝叶斯网络的精确或近似推断都是 NP 难题。目前的解决办法是使用条件独立以简化计算，或面向特定的推断要求建立简单的网络拓扑，或在不牺牲太多精确性的前提下，简化网络的结构等。虽然如此，一般仍然需要可观的计算时间。对某些问题如简单贝叶斯分类器使用条件独立则有明显的效果。

当样本数据不完全时，除了少数特例外，一般要借助于近似方法，如 Monte-Carlo 方法，Gaussian 逼近，以及 EM（期望-极大化）算法求 ML（极大似然）或 MAP（极大后验）等。尽管有成熟的算法，其计算开销也是比较大的。

6.5.3 学习贝叶斯网络的网络结构

当不能确定贝叶斯网络的结构时，用贝叶斯方法从给定数据学习网络的结构和概率分布也是可能的。由于数据挖掘面对的是大量数据，一时往往难以断定变量之间的关系，因此这个问题更具有现实意义。

首先假定表示变量集合 \mathbf{X} 的物理联合概率分布的网络结构是可以改进的。按照贝叶斯方法，定义一个离散变量表示我们对于网络结构的不确定性，其状态对应于可能的网络结构假设 S^h ，并赋予先验概率分布 $p(S^h)$ 。给定随机样本 D ， D 来自 \mathbf{X} 的物理概率分布。然后计算后验概率分布 $p(S^h|D)$ 和 $p(\theta_S|D, S^h)$ ，其中 θ_S 是参数向量，并使用这些分布回过头来计算感兴趣的期望值。

$p(\theta_S|D, S^h)$ 的计算方法与上一节类似。 $p(S^h|D)$ 的计算至少在原理上是简单的。根据贝叶斯定理有：

$$\begin{aligned} p(S^h|D) &= p(S^h, D) / p(D) \\ &= p(S^h) p(D|S^h) / p(D) \end{aligned} \quad (6.40)$$

其中 $p(D)$ 是一个与结构无关的正规化常数， $p(D|S^h)$ 是边界似然。于是确定网络结构的后验分布只需要为每一个可能的结构计算数据的边界似然。

在无约束多项分布、参数独立、采用 Dirichlet 先验和数据完整的前提下，参数向量 θ_{ij} 可以独立地更新。数据的边界似然正好等于每一个 i - j 对的边界似然的乘积：

$$p(D|S^h) = \prod_{i=1}^n \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \cdot \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})} \quad (6.41)$$

该公式首次为 Cooper 和 Herskovits 于 1992 年给出[Cooper 1992]。

在一般情况下， n 个变量的可能的网络结构数目大于以 n 为指数的函数。逐一排除这些假设是很困难的。可以使用两个方法来处理这个问题：“模型选择”方法和“选择模型平均”方法。前一个方法是从所有可能的模型（结构假设）中选择一个“好的”模型，并把它当作正确的模型。后一个方法是从所有可能的模型中选择合理数目的“好”模型，并认为这些模型代表了所有情况。问题是如何决定一个模型是否“好”？如何搜索好的模型？这些方法应用于贝叶斯网络结构能否得到准确的结果？关于“好模型”已有一些不同的定义和相应的计算方法。后两个问题要从理论上回答是困难的。然而若干研究者工作表明，使用贪婪搜索法选择单个好的假设通常会得到准确的预测[Chickering, Heckerman 1996]。使用 Monte-Carlo 方

法进行模型平均有时也很有效，甚至可以得到更好的预测。这些结果多少可以算是对目前用贝叶斯网络进行学习的巨大兴趣的回答。

Heckerman 于 1995 年提出，在参数独立、参数模块性、似然等价、以及机制独立、部件独立等假设成立的前提下，可以将学习贝叶斯非因果网络的方法用于因果网络的学习。1997 年又提出在因果马尔科夫条件下，可以由网络的条件独立和条件相关关系推断因果关系 [Heckerman 1997]。这使得在干涉（扰动）出现时可以预测其影响。

Heckerman 等使用贝叶斯网络进行数据采掘和知识发现。数据来自华盛顿高级中学的 10318 名高年级学生 [Sewell 和 Shah 1968]。每个学生用下列变量及其相应的状态来描述：

- 性别（SEX）：男、女
- 社会经济状态（SES）：低、中下、中上、高
- 智商（IQ）：低、中下、中上、高
- 家长的鼓励（PE）：低、高
- 升学计划（CP）：是、否

目标是从数据中发现影响高中生上大学意向的因素，即存在于这些变量之间的可能的因果关系。数据已经整理成如表 6.3 的充分统计。表 6.3 中每个数据表示：对于五个变量取值的某种组合（构成）统计所得到的人数。例如，第一个数据表示对（SEX=男，SES=低，IQ=低，PE=低，CP=是）这种组合统计得到的人数为 4 人。第二个数据则表示对（SEX=男，SES=低，IQ=低，PE=低，CP=否）这种组合统计得到的人数为 349 人。其后的数据依次表示轮换每个变量可能的状态统计得到的人数。变量依照从右到左的顺序轮换，状态则按照上面列出各变量的状态的顺序轮换。如此等等，依次类推。表 6.3 中前 4 行是对男生的统计数据，后 4 行是对女生的统计数据。

表 6.3 充分统计表 (人)

(男)	4	349	13	64	9	207	33	72	12	126	38	54	10	67	49	43
	2	232	27	84	7	201	64	95	12	115	93	92	17	79	119	59
	8	166	47	91	6	120	74	110	17	92	148	100	6	42	198	73
	4	48	39	57	5	47	132	90	9	41	224	65	8	17	414	54
(女)	5	454	9	44	5	312	14	47	8	216	20	35	13	96	28	24
	11	285	29	61	19	236	47	88	12	164	62	85	15	113	72	50
	7	163	36	72	13	193	75	90	12	174	91	100	20	81	142	77
	6	50	36	58	5	70	110	76	12	48	230	81	13	49	360	98

分析数据时先假定没有隐藏变量。为了生成网络参数的先验，使用容量为 5 的等价样本和一个带有一致的 $p(\mathbf{X}|S_c^h)$ 的先验网络。除了已排除的 SEX 和 SES 有父节点、CP 有子节点的结构之外，假定所有网络结构都同等相似。因为数据集是完整的，可以用 (6.40) 和 (6.41) 式计算网络结构的后验概率。通过对所有网络结构的穷举搜索，发现的两个最相似的网络结构如图 6.1。请注意，最相似的结构的后验概率也是极端接近的。如果采纳因果马尔科夫假设，并假设没有隐藏变量，则在两个图中的弧都可以有因果的解释。其中一些结果，例如社会经济状态和智商对升学愿望的影响，并不使人意外。另一些结果更有趣：从两个图中都可以得到性别对升学愿望的影响仅仅通过父母的影响体现出来。此外，两个图的不同仅仅在于 PE 和 IQ 之间的弧的方向。两个不同的因果关系似乎都有道理。右边的网络曾由 Spirtes 等用非贝叶斯方法于 1993 年选出。

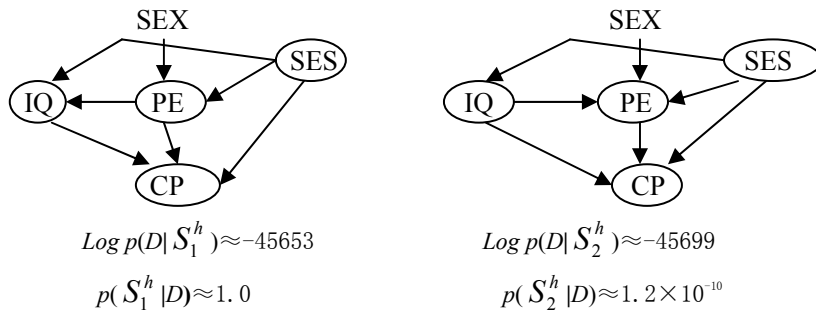


图 6.1 没有隐藏变量的后验最可能的网络结构

最值得怀疑的结果是：社会经济状况对智商有直接的影响。为了考证这个结果，考虑一个新的模型，即将图 6.1 中原来模型的直接影响用一个指向 SES 和 IQ 的隐藏变量代替。此外还考虑这样的模型，隐藏变量指向 SES, IQ 和 PE，而且在 SES—PE 和 PE—IQ 两个连接中分别去掉两个、一个和 0 个，对每个结构将隐藏变量的状态数从 2 变到 6。

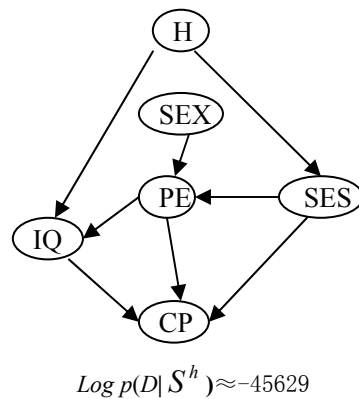


图 6.2 带有隐藏变量的后验最可能的网络结构

使用 Laplace 逼近的 Cheeseman-Stutz 变体计算这些模型的后验概率。为了找最大后验构成 $\tilde{\theta}$ s，使用 EM 算法，并在带有不同的随机初始化的 θ_s 的 100 次运行中取最大局部极大。这些模型中带有最高后验概率的一个如图 6.2。这个模型的可能性比不含有隐藏变量的最好模型高 2×10^{10} 倍。另一个最有可能的模型包含一个隐藏变量，并有一条从隐藏变量到 PE 的弧。这个模型的可能性比最好模型只差 5×10^{-9} 倍。假定没有忽略合理的模型，那么有强烈的证据表明：有一个隐藏变量在影响着 SES（社会经济状态）和 IQ（智商）。分析图 6.2 的概率可知，隐藏变量对应于“家长的素质”。

使用贝叶斯方法从先验信息和样本信息学习贝叶斯网络的结构和概率分布，进而建立贝叶斯网络，为贝叶斯网络在数据采掘和知识发现中的应用开辟了道路。与其它用于数据采掘的表示法如规则库、决策树、人工神经网络相比，贝叶斯网络有如下特点：

(1) 可以综合先验信息和后验信息，既可避免只使用先验信息可能带来的主观偏见，和缺乏样本信息时的大量盲目搜索与计算，也可避免只使用后验信息带来的噪音的影响。只要合理地确定先验，就可以进行有效的学习，这在样本难得或者代价高昂时特别有用；

(2)适合处理不完整数据集问题;

(3)可以发现数据间的因果关系。后两点在实际问题中经常遇到,而且是用其他模型难以处理的;

(4)有成熟有效的算法。虽然任意贝叶斯网络的概率推断是 NP 难题,但是很多问题加上一些限制后计算就可以简化。有些问题有近似解法。

不过,贝叶斯网络的计算量较大,在某些其它方法也可以解决的问题求解中显得效率较低。先验密度的确定虽然已经有一些方法,但对具体问题要合理确定许多变量的先验概率仍然是一个较困难的问题,而这在样本难得时却特别重要。此外,贝叶斯网络需要多种假设为前提,如何判定某个实际问题是否满足这些假设,没有现成的规则,这给实际应用带来困难。这些都是需要进一步研究的问题。尽管如此,可以预见,在数据采掘和知识发现中,尤其在具有概率统计特征的数据采掘中,贝叶斯网络将成为一个有力的工具。

6.6 贝叶斯潜在语义模型

随着互联网的普及,网上信息正在呈指数级增长趋势。合理地组织这些信息,以便从茫茫的数据世界中,检索到期望的目标;有效地分析这些信息,以便从浩如烟海的信息海洋中,挖掘出新颖的、潜在有用的模式,正在成为网上信息处理的研究热点。网上信息的分类目录组织是提高检索效率和检索精度的有效途径,如在利用搜索引擎对网页数据进行检索时,如能提供查询的类别信息,必然会缩小与限制检索范围,从而提高查准率,同时,分类可以提供信息的良好组织结构,便于用户进行浏览和过滤信息。很多大型网站都采用这种组织方式,如 Yahoo 采用人工方式来维护网页的目录结构;Google 网站采用一定的排序机制,使与用户最相关的网页排在前面,便于用户浏览。Deerwester 等人利用线性代数的知识,通过矩阵的奇异值分解(Singular Value Decomposition SVD)来进行信息滤波和潜在语义索引(Latent Semantic Index LSI)[Deerwester 1990]。它将文档在向量空间模型(VSM)中的高维表示,投影到低维的潜在语义空间(LSS)中,这一方面缩小了问题的规模,另一方面也从一定程度上避免了数据的过分稀疏现象。它在语言建模、视频检索及蛋白质数据库等实际应用中取得较好的效果。

聚类分析是文本挖掘的主要手段之一。它的主要作用是:1)通过对检索结果的聚类,将检索到的大量网页以一定的类别提供给用户,使用户能快速定位期望的目标;2)自动生成分类目录;3)通过相似网页的归并,便于分析这些网页的共性。K-均值聚类是比较典型的聚类算法,另外自组织映射(SOM)神经网络聚类和基于概率分布的贝叶斯层次聚类(HBC)等新的聚类算法也正在不断的研制与应用中。然而这些聚类算法大部分是一种无监督学习,它对解空间的搜索带有一定的盲目性,因而聚类的结果一定程度上缺乏语义特征;同时,在高维情况下,选择合适的距离度量标准变得相当困难。而网页分类是一种监督学习,它通过一系列训练样本的分析,来预测未知网页的类别归属。目前已有许多有效的算法来实现网页的分类,如 Naive Bayesian、SVM 等。遗憾的是获得大量的、带有类别标注的样本的代价是相当昂贵的,而这些方法只有通过大规模的训练集才能获得较高精度的分类效果。此外由于在实际应用当中,分类体系常常是不一致的,为目录的日常维护带来了一定的困难。Kamal Nigam 等人提出从带有类别标注和不带有类别标注的混合文档中分类 Web 网页,它只需要部分带有类别标注的训练样本,结合未标注样本含有的知识来学习贝叶斯分类器[Nigam 1998]。

我们对这一问题处理的基本思想是:如果知道一批网页 $D = \{d_1, d_2, \dots, d_n\}$ 是关于某些潜在类别主题变量 $Z = \{z_1, z_2, \dots, z_k\}$ 的描述。通过引入贝叶斯潜在语义模型,首先将含有潜在

类别主题变量的文档分配到相应的类主题中。接着利用简单贝叶斯模型，结合前一阶段的知识，完成对未含类主题变量的文档作标注。针对这两阶段的特点，我们定义了两种似然函数，并利用 EM 算法获得最大似然估计的局部最优解。这种处理方法一方面克服了非监督学习中对求解空间搜索的盲目性；另一方面它不需要对大量训练样本的类别标注，只需提供相应的类主题变量，把网站管理人员从繁琐的训练样本的标注中解脱出来，提高了网页分类的自动性。为了与纯粹的监督与非监督学习相区别，称这种方法为半监督学习算法。

潜在语义分析（LSA: Latent Semantic Analysis）的基本观点是：把高维的向量空间模型（VSM）表示中的文档映射到低维的潜在语义空间中。这个映射是通过将项/文档矩阵 $N_{m \times n}$ 的奇异值分解（SVD）来实现的。具体地说，对任意矩阵 $N_{m \times n}$ ，由线性代数的知识可知，它可以分解为下面的形式：

$$N = U \Sigma V^T \quad (6.42)$$

这里： U, V 是正交阵（ $UU^T = VV^T = I$ ）。 $\Sigma = \text{diag}(a_1, a_1, \dots, a_k, \dots, a_v)$ （ a_1, a_2, \dots, a_v 为 N 的奇异值）是对角阵。潜在语义分析通过取 k 个最大的奇异值，而将剩余的值设为零来近似（6.42）：

$$\tilde{N} = U \tilde{\Sigma} V^T \approx U \Sigma V^T = N \quad (6.43)$$

由于文档之间的相似性，可以通过 $NN^T \approx \tilde{N} \tilde{N}^T = U \tilde{\Sigma}^2 U^T$ 来表示，因此文档在潜在语义空间中的坐标可以通过 $U \tilde{\Sigma}$ 来近似。所以高维空间中的文档表示投影到低维的潜在语义空间中，原来在高维中比较稀疏的向量表示在潜在语义空间中变得不再稀疏。这也暗指，即使两篇文档没有任何共同的项，仍然可能找到它们之间比较有意义的关联值。

通过奇异值分解，将文档在高维向量空间模型中的表示，投影到低维的潜在语义空间中，有效地缩小了问题的规模。潜在语义分析在信息滤波、文本索引、视频检索等方面具有较为成功的应用。然而矩阵的 SVD 分解对数据的变化较为敏感，同时缺乏先验信息的植入等而显得过分机械，从而使它的应用受到了一定的限制。

经验表明，人们对任何问题的描述都是围绕某一主题展开的。各个主题之间具有相对明显的界限，同时由于偏爱、兴趣等的不同，对不同的主题的关注也存在着差别，也就是说对不同的主题具有一定的先验知识。基于此，我们给出文档产生的潜在贝叶斯语义模型：

设文档集合为： $D = \{d_1, d_2, \dots, d_n\}$ ，词汇集为： $W = \{w_1, w_2, \dots, w_m\}$ ，则文档 $d \in D$ 的产生模型可表述为：

- (1) 以一定的概率 $P(d)$ 选择文档 d ；
- (2) 选取一个潜在的类主题 z ，该类主题具有一定的先验知识 $p(z | \theta)$
- (3) 类主题 z 含有文档 d 的概率为： $p(z | d, \theta)$
- (4) 在类主题 z 的条件下，产生词 $w \in W$ ，其概率为： $p(w | z, \theta)$

经过上述过程获得观测点对 (d, w) ，潜在的类主题 z 被忽略掉，产生下面的联合概率模型：

$$p(d, w) = p(d)p(w|d) \quad (6.44)$$

$$p(w|d) = \sum_{z \in Z} p(w|z, \theta)p(z|d, \theta) \quad (6.45)$$

该模型是建立在下面的独立性假定下的混合概率模型。

- (1) 每一观测点对 (d, w) 的产生是相对独立的，它们通过潜在类主题相联系；
- (2) 词 w 的产生独立于具体的文档 d ，而只依赖于潜在的类主题变量 z 。

(6.45) 也表明，在某一文档 d 中，词 w 的分布是它在潜在类主题下的凸组合，组合权重是该文档类属于此主题的概率。图 6.3 表明了该模型各分量间的关联。

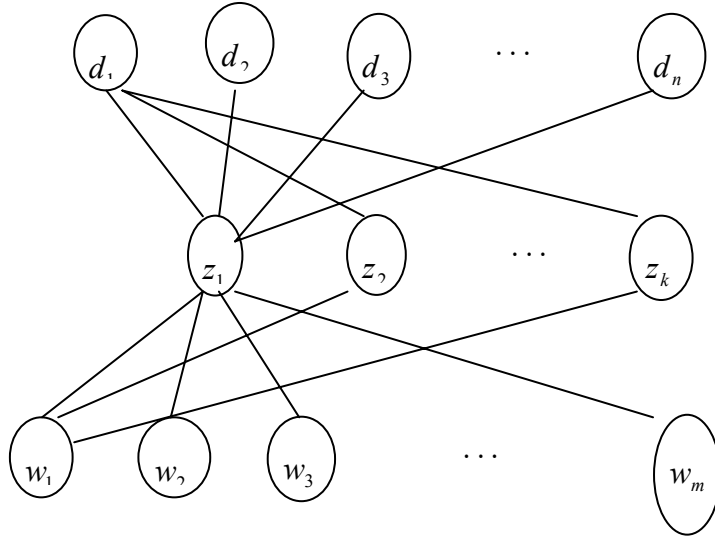


图 6.3 贝叶斯潜在语义模型

利用贝叶斯公式，并将 (6.45) 代入 (6.44) 得到：

$$p(d, w) = \sum_{z \in Z} p(z|\theta)p(w|z, \theta)p(d|z, \theta) \quad (6.46)$$

与潜在语义分析相比，贝叶斯潜在语义模型具有较为稳固的统计学基础，克服了 LSA 中的数据敏感性；它对潜在的类变量植入先验信息，避免了 SVD 的机械性。对应于 (6.42) 的奇异值分解：

$$U = \{p(d_i | z_k)\}_{n \times k} \quad V = \{p(w_i | z_k)\}_{m \times k} \quad \sum = \text{diag}(p(z_1), p(z_2), \dots, p(z_k))$$

因此 Bayes 潜在语义模型与 SVD 在形式上是统一的。

在 LSA 中，选取近似 (6.42) 的标准是最小二乘意义下的损失最小。而从贝叶斯的观点，拟合模型 (6.46) 有两种标准：最大后验概率 (MAP: Maximum A Posterior) 和最大似然估计 (ML: Maximize Likelihood)。

最大后验概率估计是在文档集 D 和词汇集 W 的情况下，使得潜在变量的后验概率最大，即：

$$P(Z | D, W) = \prod_{z \in Z} \prod_{d \in D} \prod_{w \in W} p(z | d, w) \quad (6.47)$$

由贝叶斯公式得：

$$p(z | d, w) = \frac{p(z)p(w | z)p(d | z)}{\sum_{z \in Z} p(z)p(w | z)p(d | z)} \quad (6.48)$$

最大似然估计是最大化下面的表达式：

$$\prod_{d \in D} \prod_{w \in W} p(d, w)^{n(d, w)} \quad (6.49)$$

这里 $n(d, w)$ 表示词 w 在文档 d 中出现的次数。在实际计算中一般取它的对数形式，称为对数似然。

$$\sum_{d \in D} \sum_{w \in W} n(d, w) \log p(d, w) \quad (6.50)$$

获得上述两种估计最大化的一般方法是期望最大化（EM: Expectation Maximum）方法，在 6.7 中将进行详细讨论。

一般的检索系统都是通过简单的关键词匹配算法，采用一定的排序机制，将检索到的相关网页以一定的方式提供给用户。这种处理方法的缺点是忽略了特征间的相关性，如用户查询关键词“计算机”时，传统的关键词匹配算法返回的将全是含有“计算机”的网页，而大部分关于“电脑”的网页将不能检索到，即所谓的同义词与多义词现象。从语义角度上讲，同义词指的是在不同的场景下，不同的词表达的意义相同；多义词指的是同一个词在不同的场景下表达的意义不同。而概率意义上的同义词与多义词则具有更为广泛的含义：同义词指的是在某一主题下，它们的概率关联较为密切；多义词则指的是同一个词关联着不同的类别主题。贝叶斯概率语义模型有效地解决了这一问题。若 $p(w | z)$ 较大，我们称词 w 与 z 为同义词；若对词 w 存在两个潜在类别变量 z_1, z_2 ，使得 $p(w | z_1) = p(w | z_2)$ 则称该词为多义词。下面是我们利用贝叶斯潜在语义模型，对 1199 篇足球类的文章，引入 14 个潜在类别变量，得到的部分同义词实验结果(表 6.3)。

表 6.3 基于贝叶斯潜在语义下的同义词表

泰山	申花	实德	天津泰达	深圳平安	全兴	英格兰	意大利	世界杯
泰山 山东 鲁能泰山 宿茂臻 舒畅 卡西 谢尔盖	申花 上海 彼得洛维奇 阵容 阵地 组队	实德 大连实德 王鹏 王涛 王健 孙继海 郝海东	天津泰达 泰达 金志扬 津门 张效瑞 张开 张明	深圳平安 深圳 张军 违反 违纪 刘建 李毅	四川 全兴 助攻 米罗西 马明宇 刘成 里奇	英格兰 英超 英超联赛 英德 伊斯坦布尔 水晶宫	意大利 意甲 尤文图斯 罗纳尔多 里瓦尔多 雷东多 劳尔	世界杯 足球界 足球场 预选赛 邀请赛 锦标赛 亚洲杯

桑特拉奇 拉尔	守门员 门将		张凤	李东国	黎兵 卡纳瓦罗 卡伦 安德森	欧锦赛 欧洲杯 欧洲足联	拉齐奥 拉科鲁尼 亚	亚足联 奥林匹克 国际足联
------------	-----------	--	----	-----	-------------------------	--------------------	------------------	---------------------

概率语义上的同义词与多义词的提出，不仅可以实现检索过程中同一类别之间的联想转换和不同类别之间的漫游，而且为基于贝叶斯潜在语义的半监督学习提供了依据，那就是通过潜在类别变量的引入，将含有与潜在变量同义词较多的文章聚为一类。下面将详细讨论这种算法的实现。

6.7 半监督文本挖掘算法

6.7.1 网页聚类

目前已经有很多算法来实现文本分类，获得了较高的精度（Precision）和召回率（Recall）。然而获得分类中带有标注的训练样本的代价是相当昂贵的，因而 Nigam 等人提出从带有和不带有类别标注的混合文档中学习分类 web 网页，并获得了很好的精度，但它仍然需要一定量的标注样本[Nigam 98]。网页聚类通过一定的相似性度量，将相关网页归并到一类，它也能达到缩小搜索空间的目的，然而传统的聚类方法，在处理高维和海量数据时，它的效率和精确度却大打折扣。这一方面由于非监督学习对解空间的搜索本身具有一定的盲目性，另一方面，在高维的情况下，很难找到比较适宜的相似性度量标准，例如，欧式距离度量在维数较高时，变得不再适用。基于上面的监督学习与非监督学习的特性，我们提出了一种半监督学习算法。在贝叶斯潜在语义模型的框架下，由用户提供一定数量的潜在类别变量，而不需要任何带有类别标注的样本，将一组文档集划分到不同的类别中。

它的一般模型可以描述为：已知文档集 $D = \{d_1, d_2, \dots, d_n\}$ 和它的词汇集 $W = \{w_1, w_2, \dots, w_m\}$ ，一组带有先验信息 $\theta = \{\theta_1, \theta_2, \dots, \theta_k\}$ 的潜在类别变量 $Z = \{z_1, z_2, \dots, z_k\}$ ，找出 D 上的一个划分 $D_j (j \in (1 \dots k))$ ，使得：

$$\bigcup_{j=1}^k D_j = D, \quad D_i \cap D_j = \emptyset \quad (i \neq j)$$

首先我们将 D 划分为两个集合： $D = D_L \cup D_U$ ，满足：

$$D_L = \{d \mid \exists j, z_j \in d, j \in [1 \dots k]\}, \quad D_U = \{d \mid \forall j, z_j \notin d, j \in [1 \dots k]\}$$

我们的算法对文档的类别标注分两个阶段实现：

第一阶段，对 D_L 中的元素，我们利用贝叶斯潜在语义模型，在基于 EM 参数估计的基础

上，用潜在类别变量来标注文档。即：

$$l(d) = z_j = \max_i \{p(d | z_i)\} \quad (6.51)$$

第二阶段，对 D_U 中的元素，根据对 D_L 中的元素的类别标注，利用 Naïve Bayes 分类模型，经过 EM 算法，来实现类别标注。

6.7.2 对含有潜在类别主题词的文档的类别标注

理想的情况下，任何文档都不含有两个以上的潜在类别主题词，此时只需将该文档标以潜在的类别。然而在实际应用中，这种理想的情况很难达到，一方面由于很难选择这样的潜在类别主题词，另一方面，这种要求也是不太现实的，因为两个不同的类别很可能含有多个潜在的类别主题词。如在“经济”类的文档中，很可能含有像“政治”、“文化”等词。我们的处理方法是把它分到与潜在的类别主题词语义最为密切的类别中。在我们选择的似然标准下，通过一定次数的 EM 迭代，最后通过式 (6.51) 来决定文档的类别。

EM 算法是稀疏数据参数估计的主要方法之一。它交替地执行 E 步和 M 步，以达到使似然函数值增加的目的。它的一般过程可描述为：

- (1) E 步，基于当前的参数估计，计算它的期望值；
- (2) M 步，基于 E 步参数的期望值，最大化当前的参数估计；
- (3) 对修正后的参数估计，计算似然函数值，若似然函数值达到事前制定的阈值或者指定的迭代次数，则停止，否则转(1)。

在我们的算法中，采用下面两步来实现迭代。

- (1) 在 E 步，通过下面的贝叶斯公式来获得期望值。

$$P(z | d, w) = \frac{p(z)p(d | z)p(w | z)}{\sum_{z'} p(z')p(d | z')p(w | z')} \quad (6.52)$$

从概率语义上讲，它是用潜在类别变量 z 来解释词 w 在文档 d 中出现的概率度量。

- (2) 在 M 步中，利用上一步的期望值，来重新估计参数的分布密度。

$$p(w | z) = \frac{\sum_d n(d, w)p(z | d, w)}{\sum_{d, w'} n(d, w')p(z | d, w')} \quad (6.53a)$$

$$p(d | z) = \frac{\sum_w n(d, w)p(z | d, w)}{\sum_{d', w} n(d', w)p(z | d', w)} \quad (6.53b)$$

$$p(z) = \frac{\sum_{d, w} n(d, w)p(z | d, w)}{\sum_{d, w} n(d, w)} \quad (6.53c)$$

相对于潜在语义分析中的 SVD 分解，EM 算法具有线性的收敛速度，并且简单，容易实现，可以使似然函数达到局部最优。图 6.4 是我们在实验过程中得到的 EM 迭代次数与似然函数的值的关系。

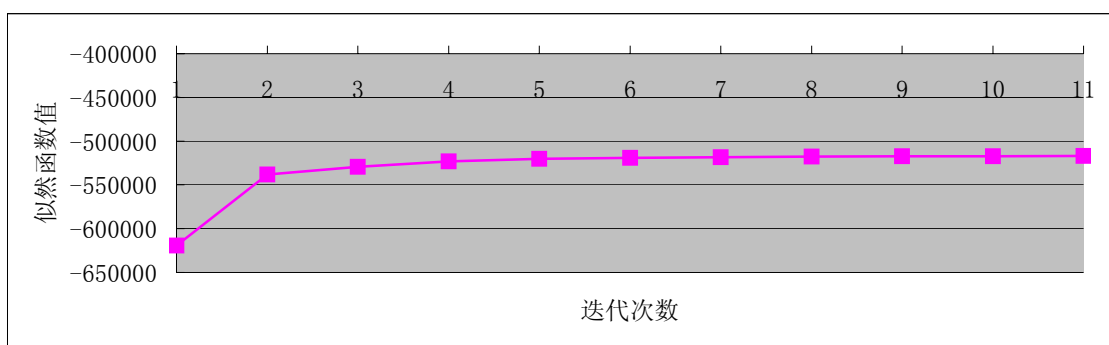


图 6.4 最大似然估计值与 EM 迭代次数的关系

6.7.3 基于简单贝叶斯模型学习标注和未标注样本

传统的分类方法都是通过一定的学习机制，对带有类别标签的训练样本学习的基础上，来决定未知样本的类别标签。然而获得大量的带有类别标注的训练样本是相当繁琐的任务。Kamal Nigam 等人研究表明，未带类别标注的文档仍然含有学习分类模型的大量信息。基于此，我们利用简单贝叶斯（Naïve Bayesian）模型作为分类器，把未带标签的训练样本作为一种特殊的缺值状态，通过一定的 EM 迭代算法来估计这种缺值。

在这里，我们首先给出 Naïve Bayes 学习文本分类的一般原理：已知训练文档集 $D = \{d_1, d_2, \dots, d_n\}$ 和它的词汇集 $W = \{w_1, w_2, \dots, w_m\}$ 。每一训练样本是一 $m+1$ 维向量：

$d_i = \langle w_1 \ w_2 \ \dots \ w_m \ c_i \rangle$ ，其中 $c_i \in C = \{c_1, c_2, \dots, c_k\}$ 是类别变量。分类的任务就是对未知类

别的样本 $d = \langle w_1 \ w_2 \ \dots \ w_m \rangle$ ，来预测它的类别：

$$c = \max_{j \in [1..k]} \{p(c_j | d, \theta)\} \quad (\text{这里 } \theta \text{ 是模型的参数})$$

为计算上式，将其展开得到：

$$p(d | c_j, \theta) = p(d) \prod_{k=1}^{|d|} p(w_k | c_j; \theta; w_q, q < k) \quad (6.54)$$

Naïve Bayes 模型在计算（6.54）时，引入了下面一些独立性假设：

- (1) 文档的词的产生独立于它的内容，即词在文档中出现的位置无先后关系。
- (2) 文档中各个词相对于类别属性是相对独立的。

在上面的独立性假定下，结合贝叶斯公式（6.54）可记为：

$$\begin{aligned}
p(d | c_j, \theta) &= p(d) \prod_{r=1}^{|d|} p(w_r | c_j; \theta) = \frac{p(c_j | \theta) p(d | c_j, \theta)}{p(d | \theta)} \\
&= \frac{p(c_j | \theta) \prod_{r=1}^m p(w_r | c_j; \theta)}{\sum_{i=1}^k p(c_i | \theta) \prod_{r=1}^m p(w_r | c_i; \theta)}
\end{aligned} \tag{6.55}$$

学习的任务变为从数据中，利用一定的先验信息来学习模型的参数。在这里我们选用多项分布模型和 Dirichlet 共轭先验。

$$\theta_{c_j} = p(c_j | \theta) = \frac{\sum_{i=1}^{|D|} I(c(d_i) = c_j)}{|D|} \tag{6.56a}$$

$$\theta_{w_t | c_j} = p(w_t | c_j; \theta) = \frac{\alpha_j + \sum_{i=1}^{|D|} n(d_i, w_t) I(c(d_i) = c_j)}{\alpha_0 + \sum_{k=1}^m \sum_{i=1}^{|D|} n(d_i, w_k) I(c(d_i) = c_j)} \tag{6.56b}$$

这里： $\alpha_0 = \sum_{i=1}^k \alpha_i$ 为模型的超参数。函数 $c(\cdot)$ 是类别标注函数， $I(a = b)$ 为示性函数（若 $a = b$ ，则 $I(a = b) = 1$ ，否则 $I(a = b) = 0$ ）。

尽管简单贝叶斯对模型的适用条件作了较为苛刻的限制，然而大量实验表明，即使在违背这些独立性假定的条件下，它仍能表现出相当的健壮性，它已经成为文本分类中广为使用的一种方法。

下面我们将通过引入一种最大化后验概率（MAP）似然标准，结合未标注样本的知识为这些未标注的样本贴标签。

考虑所有的样本集 $D = D_L \cup D_U$ ，其中 D_L 中的元素在第一阶段已被贴上标签。假设 D 中各样本的产生是相互独立的，那么下面的式子成立：

$$p(D | \theta) = \prod_{d_i \in D_U} \sum_{j=1}^{|C|} p(c_j | \theta) p(d_i | c_j; \theta) \cdot \prod_{d_i \in D_L} p(c(d_i) | \theta) p(d_i | c(d_i); \theta) \tag{6.57}$$

在上面的式子中，将未标注的文档看作是混合模型。我们的学习任务仍然是通过样本集 D 来获得模型参数 θ 的最大估计。利用贝叶斯定理：

$$p(\theta | D) = \frac{p(\theta) p(D | \theta)}{P(D)} \tag{6.58}$$

对固定的样本集而言， $p(\theta)$ 和 $p(D)$ 是常量。对（6.58）取对数得：

$$\begin{aligned}
l(\theta | D) = \log p(\theta | D) &= \log \frac{p(\theta)}{p(D)} + \sum_{d_i \in D_U} \log \sum_{j=1}^{|C|} p(c_j | \theta) p(d_i | c_j; \theta) \\
&\quad + \sum_{d_i \in D_L} \log p(c(d_i) | \theta) p(d_i | c(d_i); \theta)
\end{aligned} \tag{6.59}$$

为估计未标注样本的标签，借用潜在语义分析中的潜在变量，我们在这里引入 k 个潜在变量 $Z = \{z_1, z_2 \cdots z_k\}$ ，每个潜在变量是 n 维向量 $z_i = \langle z_{i1}, z_{i2}, \dots, z_{in} \rangle$ ，并且：如果 $c(d_j) = c_i$

那么 $z_{ij} = 1$ ，否则 $z_{ij} = 0$ 。所以 (6.59) 可以统一表示成下面的形式：

$$l(\theta | D) = \log \frac{p(\theta)}{p(D)} + \sum_{i=1}^{|D|} \sum_{j=1}^{|C|} z_{ji} \log p(c_j | \theta) p(d_i | c_j; \theta_j) \quad (6.60)$$

在 (6.59) 中，对已标注的样本 z_{ji} 是已知的，学习的任务是最大化模型的参数和对未知的 z_{ji} 的估计。

在这里，我们仍然用 EM 算法来学习未标注样本的知识。但它的过程与前一阶段有所不同。在 E 步的第 k 次迭代中，基于当前的参数估计，利用简单贝叶斯分类器来计算未标注样本的类别。对 $\forall d \in D_u$ ：

$$p(d | c_j, \theta^k) = \frac{p(c_j | \theta^k) \prod_{r=1}^m p(w_r | c_j; \theta^k)}{\sum_{i=1}^k p(c_i | \theta^k) \prod_{r=1}^m p(w_r | c_i; \theta^k)} \quad j \in [1 \dots k]$$

取获得最大后验概率的类别 c_i 作为该文档的期望类别标注。即

$$z_{id} = 1, \quad z_{jd} = 0 \quad (j \neq i)$$

在 M 步，基于前一步获得的期望值，最大化当前的参数估计：

$$\theta_{c_j} = p(c_j | \theta) = \frac{\sum_{i=1}^{|D|} z_{ji}}{|D|} \quad (6.61a)$$

$$\theta_{w_i | c_j} = p(w_i | c_j; \theta) = \frac{\alpha_j + \sum_{i=1}^{|D|} n(d_i, w_i) z_{ji}}{\alpha_0 + \sum_{k=1}^m \sum_{i=1}^{|D|} n(d_i, w_k) z_{ji}} \quad (6.61b)$$

4) 实验设计与结果分析

我们的实验数据是用搜索引擎 Spider 从 <http://www.fm365.com> 搜集的关于体育方面的网页，在每一类别中都包括了含有类别词的网页和不含有类别词的网页。它们的类别和分布见表 6.4：

表 6.4 选择的训练文档及其分布

	足球	篮球	排球	乒乓球	网球	棋牌
含有主题词	40	60	48	30	57	80
不含有主题词	80	40	29	11	6	4

上述共有 485 篇网页，经过切词处理后，去掉一定的停用词后，共计有 2719 个词，经过半监督学习算法处理后，得到表 6.5 的结果：

表 6.5 半监督文本挖掘的结果

	足球	篮球	排球	乒乓球	网球	棋牌	结果评价	
							精度	召回率
足球(120)	112	5	0	1	2	2	0.96552	0.93333
篮球(100)	1	98	0	1	0	0	0.93333	0.98000
排球(77)	1	2	74	0	0	0	0.97368	0.96103
乒乓球(41)	0	0	0	40	1	0	0.83333	0.97561
网球(63)	0	0	1	4	58	0	0.93548	0.92063
棋牌(84)	4	0	1	2	1	76	0.97436	0.90476

在足球类的 120 篇文档中，分到篮球、排球、乒乓球、网球及棋牌类的文档个数分别为 5、0、1、2、2。其中分到篮球类的文档数较多，通过进一步的研究我们发现，这些文档中含有的词与篮球类中含有的词中同义词较多所致。我们使用同样的数据，利用 Naïve Bayes 分类，得到类似的结果。

另外，选取足球类的文档 1000 篇，经过初步的预处理后得到 876 个词。图 6.5 是选取不同的潜在类别变量分到各类中的文档个数对比。

第一次选择 14 个潜在变量，第二次选择 7 个潜在变量，两次的差别是在第一次选择中，“甲 A”的各个俱乐部在第二次中用“甲 A”来替代。从图中可以看出，第一次选择各个类别分到的文档数基本相同，而在第二次选择中，分到“甲 A”中的文档数近似为各“俱乐部”中的文档数之和，分到其余类别中的文档数基本不变。这个结果与我们的抽样基本是吻合的，同时也说明，潜在类别变量的概括能力具有一定的层次性。

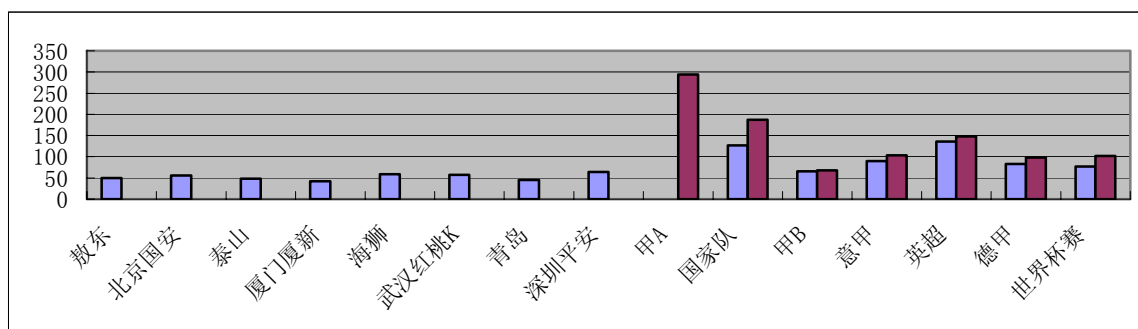


图 6.5 各类中的文档个数对比

网上信息的分类目录组织是提高检索效率和检索精度的有效途径。它通过学习大量的带有类别标注的训练样本，来预测网页的类别，然而人工标注这些训练样本是相当繁琐的。网页聚类通过一定的相似性度量，将相关网页归并到一类，它也能达到缩小搜索空间的目的，然而传统的聚类方法，对解空间的搜索带有盲目性和缺乏语义特性，因而它的效率和精确度大打折扣。我们提出的半监督学习算法，在贝叶斯潜在语义模型的框架下，由用户提供一定数量的潜在类别变量，而不需要任何带有类别标注的样本，将一组文档集划分到不同的类别中。它分为两个阶段：第一阶段，利用贝叶斯潜在语义分析来标注含有潜在类别变量的文档的类别，第二阶段则通过简单贝叶斯模型，结合未标注文档的知识，对这些文档贴标签。实验结果也表明，该算法具有较高的精度与召回率。我们将进一步研究潜在类别变量的选择对结果

的影响, 以及在贝叶斯潜在语义分析框架下如何实现词的聚类等问题。

习 题

1. 分别解释条件概率、先验概率和后验概率。
2. 描述全概率下的贝叶斯公式, 并详细说明贝叶斯公式的重要意义。
3. 描述几种常用的先验分布选取方法。
4. 简单贝叶斯分类中简单的含义是什么, 简述改进分类方法的主要思想。
5. 叙述贝叶斯网络的结构和建立方法, 试举例描述贝叶斯网络的使用。
6. 何谓半监督文本挖掘, 描述贝叶斯模型在网页聚类中的应用。
7. 近年来随着 Internet 技术的发展, 贝叶斯规则被广泛应用, 试举两个具体的贝叶斯规则应用实例, 并加以解释。

第七章 归纳学习

7.1 概 述

归纳学习是符号学习中研究得最为广泛的一种方法。给定关于某个概念的一系列已知的正例和反例,其任务是从中归纳出一个一般的概念描述。归纳学习能够获得新的概念,创立新的规则,发现新的理论。它的一般的操作是泛化 (generalization) 和特化 (specialization)。泛化用来扩展一假设的语义信息,以使其能够包含更多的正例,应用于更多的情况。特化是泛化的相反的操作,用于限制概念描述的应用范围。

用程序语言描述定义上述内容的过程就是归纳学习程序,用于书写归纳程序的语言称为归纳程序设计语言,能执行归纳程序,完成特定归纳学习任务的系统叫做归纳学习系统。归纳学习系统可独立,也可嵌入另一较大的知识处理系统。一般归纳程序的输入是科学实验中个别观察对象(过程)的描述,输出是一类对象的总体特征描述或几类对象的分类判别描述。

与演绎相对照,归纳的开始前提是具体事实而不是一般公理,推理目标是形式化解释事实的似然一般断言和预见新事实。归纳推理企图从给定现象或它的一部分的具体观察推导出一个完整的、正确的描述。归纳的两个方面——似然假设的产生和它的有效性(真值状态的建立),只有前者对归纳学习研究具备基本意义,而假设有效性的问题是次要的,因为假定所产生的假设由人类专家判断,由已知的演绎推理和数理统计的方法测试。

归纳学习可以分为实例学习、观察与发现学习。实例学习,又叫概念获取,它的任务是确定概念的一般描述,这个描述应能解释所有给定的正例并排除所有给定的反例。这些正反例由信息源提供。信息源的来源非常广泛,可以是自然现象,也可以是实验结果。实例学习是根据教师给以分类的正反例进行学习,因此是有教师学习。

观察与发现学习又称描述的泛化。这类学习没有教师的帮助,它要产生解释所有或大多数观察的规律和规则。这类学习包括概念聚类、构造分类、发现定理、形成理论等。观察与发现学习是由未经分类的观察学习,或由系统自身的功能去发现,因此是无教师学习。

因为归纳推理是从有限的、不完全的知识状态推出完全的知识状态,故归纳推理本身就是一种非单调推理。但归纳推理本身又无法验证新知识正确与否,而非单调逻辑则为我们处理非单调归纳知识提供理论基础。

归纳原理的基本思想是在大量观察的基础上通过假设形成一个科学理论。所有观察都是单称命题,而一个理论往往是领域内的全称命题,从单称命题过渡到全称命题从逻辑上来说没有必然的蕴涵关系,对于不能观察的事实往往默认它们成立。我们把归纳推理得到的归纳断言作为知识库中的知识使用,而且做为默认知识使用,当出现与之矛盾的新命题时,可以推翻原有的由归纳推理得出的默认知识,以保持系统知识的一致性。

单个概念的归纳学习的一个通用定义是:

- (1) 给定由全体实例组成的一个实例空间,每个实例具有某些属性。
- (2) 给定一个描述语言,该语言的描述能力包括描述每一个实例(通过描述改实例的属性来实现)及描述某些实例集,称为概念。
- (3) 每次学习时,由实例空间抽出某些实例,称这些实例构成的集合为正例集。再由实例空间抽出另外一些实例,称这些实例为反例集。
- (4) 如果能够在有限步内找到一个概念 A,它完全包含正例集,并且与反例集的交集为空,

则 A 就是所要学习的单个概念, 学习成功, 否则, 学习失败。

(5) 如果存在一个确定的算法, 使得对于任意给定的正例集和反例集, 学习都是成功的, 则称该实例空间在该语言表示下是可学习的。

归纳学习中具有代表性的学习方法有变型空间、AQ11 算法、决策树方法等, 本章将分别进行讨论。

7.2 归纳学习的逻辑基础

7.2.1 归纳学习的一般模式

为了较具体地刻画概念的归纳学习, 这里给出归纳学习的一般模式。

给定:

① 观察语句集(事实) F : 这是有关某类对象中个别具体对象的知识或某一对象的部分特征的知识。

② 假定的初始归纳断言(可空): 是关于目标的泛化项或泛化描述。

③ 背景知识: 背景知识定义了观察语句和所产生的候选归纳断言上的假定和限制, 以及任何有关问题领域知识。有关问题领域知识包括特化所找归纳断言的期望性质的择优标准。

寻找:

归纳断言 H (假设), H 重言或弱蕴涵观察语句并满足背景知识。

假设 H 永真蕴涵事实 F , 是指 F 是 H 的逻辑推理, 即 $H \Rightarrow F$ 成立。也就是若表达式 $H \Rightarrow F$ 在所有解释下均为真, 可表示为:

$$H \triangleright F \quad H \text{特化为 } F$$

或
$$F \vdash H \quad F \text{归结或泛化为 } H$$

这里, 从 H 推导出 F 的过程是保真过程, 因为根据上述模式 $H \Rightarrow F$ 必成立, 故只要 H 为真, F 必为真。反之, 从事实 F 推导出假设 H 的过程是保假过程, 即如果有事实 F 为假, 则 H 一定为假。

所谓 H 弱蕴涵 F , 意指事实 F 不是 H 确定的结论, 而是 H 合理的或部分的结论。有了弱蕴涵概念, 这个模式就可以有只需解释所有事实中的某些事实的可能的和部分的假设。不过我们仍然集中注意在永真蕴涵事实的假设上。

对于任意给定的事实集, 可能产生无数个蕴涵这些事实的假设, 这时就需要背景知识来提供限制条件和优先原则, 以便将假设从无穷个减少到一个假设或者几个最优的假设。

为了将概念归纳学习的逻辑基础形式化, 表 7.1 给出了基本符号, 并附上简单的解释。

表 7.1 给出了基本符号表

符 号	意 义
\sim	非
\wedge	合取 (逻辑乘)

\vee	析取（逻辑加）
\Rightarrow	蕴涵
\Leftrightarrow	逻辑等价
\leftrightarrow	项重写
\oplus	异或
F	事实集
H	假设
\triangleright	特化
\triangleright	泛化
\models	重新形式化
$\exists v_i$	存在量词约束变项 v_i
$\exists! v_i$	数值存在量词约束变项 v_i
$\forall v_i$	全称量词约束变项 v_i
D_i	概念描述
K_i	判断一个概念的名字的谓词
$::>$	将概念描述与概念名连接的蕴涵
e_i	一个事件（对一种情况的描述）
E_i	仅对概念 K_i 的事件为真的谓词
X_i	属性
LEF	评价函数
$\text{DOM}(P)$	描述符 P 的定义域

7.2.2 概念获取的条件

概念获取的一类特殊情况，它的观察语句集 F 是一个蕴涵的集合，其形式如下：

$$F: \{e_{ik} ::> K_i\} \quad i \in I \quad (7.1)$$

其中， e_{ik} (K_i 的训练事件) 是概念 K_i 的第 k 个例子的符号描述。概念的谓词 K_i ， I 是 K_i 的下标集合。 $e_{ik} ::> K_i$ 的含义是“凡符合描述 e_{ik} 的事件均可被断言为概念 K_i 的例子”。学习程序要寻求的归纳断言 H 可以用概念识别规则集来刻画，形式如下：

$$H: \{D_i ::> K_i\} \quad i \in I \quad (7.2)$$

其中 D_i 是概念 K_i 的描述, 即表达式 D_i 是事件的逻辑推论, 该事件可被断言为概念 K_i 的一个例子。

用 E_i 来表示概念 K_i 中所有训练事件的描述 ($i \in I$), 根据归纳断言的定义, 必须有 $H \vdash F$ 。要使 D_i 成为概念 K_i 的描述, 用式 (6.1) 和式 (6.2) 分别代替这里的 H 和 F , 则下述条件必须成立:

$$\forall i \in I \quad (E_i \Rightarrow D_i) \quad (7.3)$$

即所有 K_i 的训练事件必须符合 D_i 。如果约定每一训练事件只能属于一个概念, 则下面条件也成立:

$$\forall i, j \in I \quad (D_i \Rightarrow \sim E_j) \quad \text{若 } i \neq j \quad (7.4)$$

其含意是任何一个概念 K_i ($j \neq i$) 的训练事件均不符合 D_i 。

表达式条件 (6.3) 称为完整性条件, 条件 (6.4) 称为一致性条件。作为概念识别规则接受的归纳断言一定要满足这两个条件, 从而保证 D_i 的完整性和一致性。完整性和一致性条件为实例学习概念的算法提供了逻辑基础。

一类物体的特征描述是一个满足完整性条件的表达式, 或是这种表达式的合取。这种描述从所有可能的类别中判别给定类。一个物体类的差别描述则是满足完整性和一致性条件的表达式, 或这些表达式的析取, 其目标是在一定数目的其它类中标识给定类。

知识获取的主要兴趣在于面向符号描述的推导。这种描述要易于理解, 在产生它所表达信息的智能模型时要易于应用, 因此由归纳推理产生的描述要与人类的知识表示相似。

归纳学习方法分类中, 一个指导性原则是选择归纳断言语言的类型。例如, 常用谓词逻辑的某种限定形式或与此类似的概念, 或者决策树、产生式规则、语义网络、框架等方式, 或者采用多值逻辑、模态逻辑等。

在谓词逻辑的基础上, 进行修改和扩充, 增加一些附加形式和新概念, 以增强表达能力。Michalski 等提出了一种标注谓词演算 APC (Annotated Predicate Calculus), 使之更适合于归纳推理。APC 与通常谓词演算之间的主要差别是: ①每个谓词、变量和函数都被赋予一个标注。标注是该描述符学习问题有关的背景知识的集合。例如描述符所代表概念的定义, 该标注与其它概念的关系, 描述符作用范围说明等。②除谓词外, APC 还包括复合谓词, 这些复合谓词的参数可以是复合项。一个复合项是几个通常项的组合, 如 $P(t_1 \vee t_2, A)$ 。③表达项之间关系的谓词被表示为选择符关系, 如: $=, \neq, >, \geq, \leq, <$ 。④除全称量词和存在量词外, 还有数字量词。该量词用来表达满足一个表达式的物体数量信息。

7.2.3 问题背景知识

对于一个给定的观察陈述集, 有可能构造无数个蕴涵这些陈述的归纳断言。因此, 有必要使用一些附加信息, 即问题的背景知识, 限制可能的归纳断言范围, 并从中决定一个或若干个最佳的归纳断言。例如, 在 Star 的归纳学习方法中, 背景知识包括这样几部分: ①观察陈述中用到的描述符信息, 该信息附在每个描述符标注中。②关于观察和归纳断言的形式假设。③列举归纳断言应有特性的选择标准。④各种推理规则、启发式规则、特殊子程序、通用的和独立的过程, 以便允许学习系统产生给定断言的逻辑结论和新的描述符。由于观察陈述中描述符选择对产生归纳断言有重要影响, 因此先考虑描述符选择问题。

学习系统输入的主要内容是一个观察陈述集。这些陈述中的描述符是事物可观察到的特性和可利用的测量数据，决定这些描述符是归纳学习的一个主要问题。可以通过初始描述符与学习问题的联系程度来刻画学习方法。这几种联系有：①完全相关，即观察陈述集中的所有描述符都与学习任务直接相关，学习系统的任务是形成一个联系这些描述符的归纳断言。②部分相关，观察陈述集中可能有许多无用或冗余的描述符，有些描述符是相关的，此时学习系统的任务是选择出其中最相关的描述符，并用它们构造合理的归纳断言。③间接相关，观察陈述不包括与问题直接相关的描述符。但是，在初始描述中，有一些描述符可以用来生成相关的描述符。学习系统的任务是生成这些直接相关的描述符，并由此得到归纳断言。

描述符标注是描述符与学习问题有关背景知识的集合，包括：

- (1) 定义域和描述符的类型说明；
- (2) 与描述符有关的操作符说明；
- (3) 描述符之间的约束和关系说明；
- (4) 表示数量的描述符在问题中的意义、变化规律；
- (5) 描述符可应用事物的特性；
- (6) 指出包含给定描述符的类，即该描述符的父节点；
- (7) 可代替该描述符的同义词；
- (8) 描述符的定义；
- (9) 给出物体描述符的典型例子。

描述符定义域是描述符所能取值的集合。如人的体温在 $34^{\circ}\text{C} \sim 44^{\circ}\text{C}$ 之间，则描述符“体温”只能在这个范围内取值。描述符类型则是根据描述符定义域元素之间的关系决定的。根据描述符定义域的结构，有三种基本类型：

(1) 名称性描述符。这种描述符的定义域由独立的符号或名字组成，即值集中值之间没有结构关系。例如水果、人名等。

(2) 线性描述符。该类描述符值集中的元素是一个全序集。例如，资金、温度、重量、产量等都是线性描述符。表示序数、区间、比率和绝对标度的变量都是线性描述符的特例。将一个集合映射成一个完全有序集的函数也是线性描述符。

(3) 结构描述符。其值集是一个树形的图结构，反映值之间的生成层次。在这样的结构中，父节点表示比子节点更一般的概念。例如，在“地名”的值集中，“中国”是节点“北京”、“上海”、“江苏”、“广东”等的父节点。结构描述符的定义域是通过问题背景知识说明的一组推理规则来定义的。结构描述符也能进一步细分为有序和无序的结构描述符。描述符的类型对确定应用描述符的操作是很重要的。

在 Star 学习系统中，断言的基本形式是 c-表达式，定义为一个合取范式：

$$\langle \text{量词形式} \rangle \langle \text{关系陈述的合取} \rangle \quad (6.5)$$

其中， $\langle \text{量词形式} \rangle$ 表示零个或多个量词。 $\langle \text{关系陈述} \rangle$ 是特殊形式的谓词。下面是一个 c-表达式的例子：

$$\exists P_0, P_1 ([\text{shape}(P_0 \wedge P_1) = \text{box}] [\text{weight}(P_0) > \text{weight}(P_1)])$$

即物体 P_0 和 P_1 的形状都是盒子，物体 P_0 比 P_1 重。

c-表达式的一个重要的特殊形式是 a-表达式，即原子表达式，其中不含“内部析取”。所谓内部合取、析取，是指连接项的与、或；而外部合取、析取指连接谓词的与、或，即通常意义下的逻辑与、或。

7.2.4 选择型和构造型泛化规则

一个泛化规则是从一个描述到一个更一般描述之间的转换。更一般的描述永真蕴涵初始描述。由于泛化规则是保假的，即若 $F \models H$ ，则对于有事实使 F 为假，也一定会使 H 为假 ($\sim F \Rightarrow \sim H$)。

概念获取中，如果要将一个规则 $E::> K$ 转换成一个更一般的规则 $D::> K$ ，必须有 $E \Rightarrow D$ 。因此可利用形式逻辑中的永真蕴涵来获得泛化规则。如形式逻辑中有 $P \wedge Q \Rightarrow P$ ，则可将其转换成泛化规则：

$$P \wedge Q ::> K \models P ::> K \quad (7.6)$$

用标注谓词演算来表示这些泛化规则，主要考虑将一个或多个陈述转换成单个更一般陈述的泛化规则：

$$\{D_i ::> K_i\} \quad i \in I \models D ::> K \quad (7.7)$$

等价于：

$$D_1 \wedge D_2 \wedge \dots \wedge D_n ::> K \models D ::> K \quad (7.8)$$

该规则表示，如果一个事件满足所有的描述 D_i ($i \in I$)，则它也满足一个更一般的描述 D 。

泛化转换的一个基本性质是：它得出的只是一个假设，需用新的数据来测试，且泛化规则并不保证所得描述符是合理的或有用的。泛化规则分为两类：构造型和选择型。若在产生概念描述 S 中，用到的描述都是在初始概念描述 D_i ($i \in I$) 中出现过，则称为选择型泛化，否则称为构造型泛化规则。

1. 选择型泛化规则

设 CTX, CTX_1, CTX_2 代表任意的表达式。

(1) 消除条件规则

$$CTX \wedge S ::> K \models CTX ::> K \quad (7.9)$$

其中 S 是任意的谓词或逻辑表达式。

(2) 增加选择项规则

$$CTX_1 ::> K \models CTX_1 \vee CTX_2 ::> K \quad (7.10)$$

通过增加选择项将概念描述泛化，如：

$$CTX \wedge [\text{color} = \text{red}] ::> K \models CTX \wedge [L = R_2] ::> K$$

(3) 扩大引用范围规则

$$CTX \wedge [L = R_1] ::> K \models CTX \wedge [\text{color: red} \wedge \text{blue}] ::> K \quad (7.11)$$

其中 $R_1 \subseteq R_2 \subseteq \text{DOM}(L)$ ， $\text{DOM}(L)$ 为 L 的域， L 是一个项， R_2 是 L 取值的一个集合。

(4) 闭区间规则

$$\begin{array}{l} CTX \wedge [L = a] ::> K \\ CTX \wedge [L = b] ::> K \end{array} \models CTX \wedge [L = a \dots b] ::> K \quad (7.12)$$

其中 L 是线性描述符, a 和 b 是 L 的一些特殊值。

(5) 爬山泛化树规则

$$\left. \begin{array}{l} \text{CTX} \wedge [L = a] ::> K \\ \text{CTX} \wedge [L = b] ::> K \\ \vdots \\ \text{CTX} \wedge [L = i] ::> K \end{array} \right| < \text{CTX} \wedge [L = S] ::> K \quad (7.13)$$

其中 L 是结构描述符, 在 L 的泛化树域中, S 表示后继为 a, b, \dots, i 的最低的父节点。

(6) 将常量转换为变量规则

$$\left. \begin{array}{l} F[a] \\ F[b] \\ \vdots \\ F[i] \end{array} \right| < \forall x F[x] \quad (7.14)$$

其中 $F[x]$ 是依赖于变量 x 的描述符, a, b, \dots, i 是常量。对于描述 $F[x]$, 若 x 的某些值 (a, b, \dots, i) 使 $F[x]$ 成立, 则可得到假设: 对于 x 的所有值, $F[x]$ 成立。

(7) 将合取转换为析取规则

$$F_1 \wedge F_2 ::> K \quad | \quad F_1 \vee F_2 ::> K \quad (7.15)$$

其中 F_1, F_2 为任意描述。

(8) 扩充量词范围规则

$$\forall x F[x] ::> K \quad | \quad \exists x F[x] ::> K \quad (7.16)$$

$$\exists_{(I_1)} x F[x] ::> K \quad | \quad \exists_{(I_2)} x F[x] ::> K \quad (7.17)$$

其中 I_1, I_2 是量词的域 (整数集合), 且 $I_1 \subseteq I_2$ 。

(9) 泛化分解规则

用于概念获取

$$\left. \begin{array}{l} P \wedge F_1 ::> K \\ \sim P \wedge F_2 ::> K \end{array} \right| < F_1 \vee F_2 ::> K \quad (7.18)$$

用于描述泛化

$$P \wedge F_1 \vee \sim p \wedge F_2 \quad | \quad F_1 \wedge F_2 \quad (7.19)$$

这里 P 均为谓词。

(10) 反扩充规则

$$\left. \begin{array}{l} \text{CTX}_1 \wedge [L = R_1] ::> K \\ \text{CTX}_2 \wedge [L = R_2] ::> \sim K \end{array} \right| < [L \neq R_2] ::> K \quad (7.20)$$

其中 R_1, R_2 是析取式。

给定一个属于概念 K 的对象描述 (正例) 和一个不属于概念 K 的对象描述 (反例), 该规则将生成一个包括这两个描述的更一般描述。这是实例学习差别描述的一个基本规则。

2. 构造型泛化规则

构造性泛化规则能生成一些归纳断言, 这些归纳断言使用的描述符不出现在初始的观察

陈述中，也就是说，这些规则对初始表示空间进行了变换。

(1) 通用构造型规则

$$\left. \begin{array}{l} \text{CTX} \wedge F_1 ::> K \\ F_1 \Rightarrow F_2 \end{array} \right| < \text{CTX} \wedge F_2 ::> K \quad (7.21)$$

该规则表示，若一个概念描述含有一部分 F_1 ，已知 F_1 蕴涵另一概念 F_2 ，则通过用 F_2 替代 F_1 可得到一个更一般的描述。

(2) 计算变量规则

计算量词变量 CQ 规则：

$$\exists v_1, v_2, \dots, v_k F[V_1, V_2, \dots, V_k]$$

CQ 规则将产生一个新的描述符“# v -COND”，表示满足某条件 COND 的 v_i 的个数。例如，

“# v_i -length—2..4”，表示其长度在 2 和 4 之间 v_i 的个数。

计算谓词变量个数 CA 规则：在描述中一个描述符是一个具有几个变量的关系 $\text{REL}(v_1, v_2, \dots)$ ，CA 规则将计算关系 REL 中满足条件 COND 的个数。

(3) 产生链属性规则

概念描述中，若一个概念描述中传递关系不同出现的变量形成一条链，该规则能生成刻画链中某些特定对象的特征的描述符。这种对象可能是：

LST-对象：“最小的对象”，或链的开始对象。

MST-对象：链的结束对象。

MID-对象：链中间的对象。

Nth-对象：链中第 N 个位置上的对象。

(4) 检测描述符之间的相互依靠关系规则

假设已知一个例示某个概念的对象集合，用属性描述来刻画对象的特征。这些描述仅定义出对象的属性值，它们不刻画对象的结构特性。假设在所有事实描述中，线性描述符 x 的值按升序排列，而另一个线性描述符 y 的对应值也是升序或降序的，则生成一个 2 维描述符 $M(x, y)$ ，表示 x, y 之间有单调关系。当 y 值为升序时，描述符取值 \uparrow ，当降序时取值 \downarrow 。

7.3 偏置变换

偏置在概念学习中具有重要作用。所谓偏置，是指概念学习中除了正、反例子外，影响假设选择的所有因素。这些因素包括：①描述假设的语言。②程序考虑假设的空间。③按什么顺序假设的过程。④承认定义的准则，即研究过程带有已知假设可以终止还是应该继续挑选一个更好的假设。采用偏置方法，学习部分选择不同的假设，会导致不同的归纳跳跃。偏置有两个特点：

(1) 强偏置是把概念学习集中于相对少量的假设；反之，弱偏置允许概念学习考虑相对大量的假设。

(2) 正确偏置允许概念学习选择目标概念，不正确偏置就不能选择目标概念。

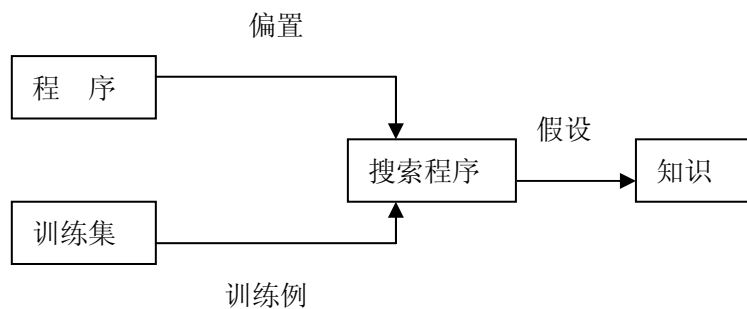


图 7.1 偏置在归纳学习中的作用

图 7.1 给出了偏置在归纳学习中的作用。由图可知，给定任何特殊顺序的训练例子，归纳就变成一个自变量的函数。当偏置很强且正确时，概念学习能立即选择可用的目标概念。当偏置很弱且不正确时，概念学习的任务是最困难的，因为没有什么引导可以选择假设。为了变换一个较弱的偏置，可以采用下面的算法：

- (1) 经过启发式，推荐新概念描述加到概念描述语言。
- (2) 变换推荐的描述成为概念描述语言已形式化表示的新概念描述。
- (3) 同化任何新概念进入假设的限定空间，保持假设空间机制。

上述算法中，第(1)步确定一个更好的偏置。第(2)、(3)步机器执行变换，其结果新概念描述语言较前面的描述语言要好。

实现归纳学习，需要研究一个好的偏置。关于偏置变换的根本问题，包括同化生成知识获取问题的任务、初始偏置的计算、目标自由与目标敏感的方法等，尚需进一步研究。

7.4 变型空间方法

变型空间(Version Space)方法以整个规则空间为初始的假设规则集合 H 。依据训练例子中的信息，它对集合 H 进行泛化或特化处理，逐步缩小集合 H 。最后使 H 收敛为只含有要求的规则。由于被搜索的空间 H 逐步缩小，故称为变型空间。

Mitchell(1977)指出，规则空间中的点之间可以按其一般性程度建立偏序关系。在图 7.2 中表示了一个规则空间中偏序关系的一部分。其中 TRUE 表示没有任何条件，这是最一般的概念。概念 $\exists x \text{ CLUBS}(x)$ 表示至少有一张梅花牌，它比前者更特殊。概念 $\exists x, y \text{ CLUBS}(x) \wedge \text{HEARTS}(y)$ 表示至少有一张梅花牌且至少有一张红心牌，它比前两个概念都特殊。图中的箭头是从较特殊的概念指向较一般的概念。

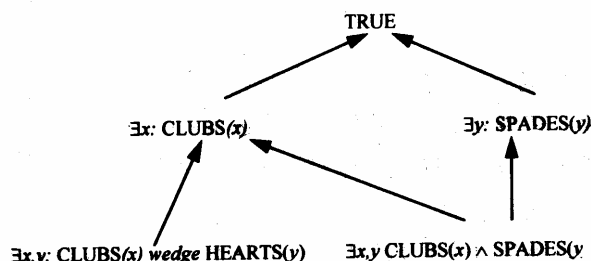


图 7.2 一个规则空间的偏序关系

一般规则空间排序后的示意图见图 7.3。图中最上面的一个点是最一般的规则（概念），是没有描述的点，即没有条件的点。所有例子都符合这一概念。图中最下面一行的各点是训练正例直接对应的概念。每个点的概念只符合一个正例。例如，每个例子都是一张牌 C 的花色和点数，一个例子是

$\text{SUIT}(C, \text{clubs}) \wedge \text{RANK}(C, 7)$

这是一个训练正例，又是一个最特殊的概念。概念 $\text{RANK}(C, 7)$ 是在规则空间中部的点。它比没有描述更特殊，但比训练正例更一般。

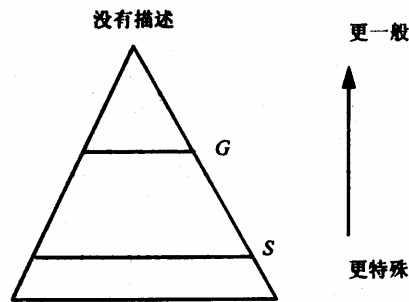


图 7.3 一般规则空间排序示意图

在搜索规则空间时，使用一个可能合理的假设规则的集合 H 。 H 是规则空间的子集，它是规则空间中间一段。 H 中最一般的元素组成的子集称为 G 集合， H 中最特殊的元素组成的子集称为 S 集合。在规则空间中， H 是上界 G 和下界 S 之间的一段。因此可以用 G 和 S 表示集合 H 。

变型空间方法的初始 G 集是最上面的一个点（最一般的概念），初始 S 集是最下面的直线上的点（训练正例），初始 H 集是整个规则空间。在搜索过程中， G 集逐步下移（进行特化）， S 集逐步上移（进行泛化）， H 逐步缩小。最后 H 收敛为只含一个要求的概念。下面分别介绍几种算法。

7.4.1 消除候选元素算法

Mitchell 的算法称为消除候选元素算法，它利用边界集合 G 和 S 表示集合 H 。集合 H 称为变型空间， H 中所有的概念描述都满足至此已提供的全部正例，且不满足至此已提供的任一反例。

开始时， H 是整个规则空间。在接受训练正例后，程序进行泛化，从 H 中去掉一些较特殊的概念，使 S 集上移。在接受训练反例后，程序进行特殊化，从 H 中去掉一些较一般的概念，使 G 集下移。二者都从 H 中消除一些候选概念。算法过程可分四步。

算法 7.1 消除候选元素算法

(1) 初始化 H 集是整个规则空间，这时 S 包含所有可能的训练正例（最特殊的概念）。这时 S 集规模太大。实际算法的初始 S 集只包含第一个训练正例，这种 H 就不是全空间了。

(2) 接收一个新的训练例子。如果是正例，则首先由 G 中去掉不覆盖新正例的概念，然后修改 S 为由新正例和 S 原有元素共同归纳出的最特殊的结果（这就是尽量少修改 S ，但要求 S 覆盖新正例）。如果这是反例，则首先由 S 中去掉覆盖该反例的概念，然后修改 G 为由

新反例和 G 原有元素共同作特殊化的最一般的结果（这就是尽量少修改 G ，但要求 G 不覆盖新反例）。

(3) 若 $G=S$ 且是单元素集，则转 (4)，否则转 (2)。

(4) 输出 H 中的概念（即 G 和 S ）。

下面给出一个实例。用特征向量描述物体，每个物体有两个特征：大小和形状。物体的大小可以是大的 (lg) 或小的 (sm)。物体的形状可以是圆的 (cir)、方的 (squ) 或三角的 (tri)。要教给程序“圆”的概念，这可以表示为 (x, cir) ，其中 x 表示任何大小。

初始 H 集是规则空间。 G 和 S 集分别是

$$G = \{(x, y)\}$$

$$S = \{(\text{sm squ}), (\text{sm cir}), (\text{sm tri}), (\text{lg squ}), (\text{lg cir}), (\text{lg tri})\}$$

初始变型空间 H 如图 7.4 所示。

第一个训练例子是正例 (sm cir)，这表示小圆是圆。经过修改 S 算法后得到

$$G = \{(x, y)\}$$

$$S = \{(\text{sm cir})\}$$

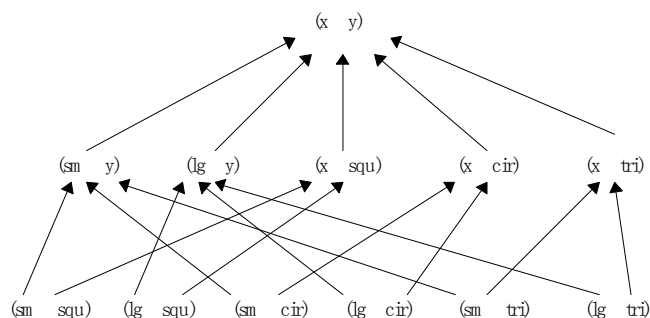


图 7.4 初始变型空间

图 7.5 显示第一个训练例子后的变型空间。图中由箭头线连接的四个概念组成变型空间。满足第一个训练例子的正是这四个概念，并仅有这四个。实际算法中以这个作为初始变型空间，而不用图 7.4。

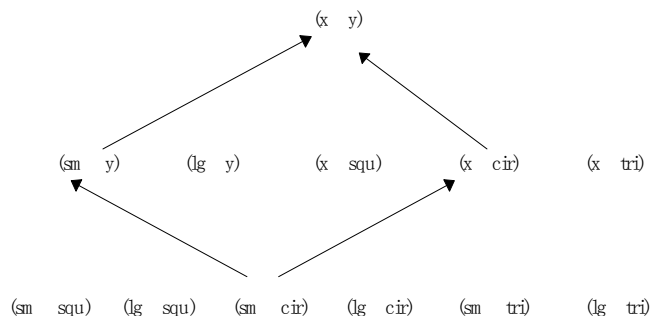


图 7.5 第一个训练例子后的变型空间

第二个训练例子是反例 (lg tri)。这表示大三角不是圆。这一步对 G 集进行特化处理，得到

$$G = \{(x, \text{cir}), (\text{sm}, y)\}$$

$$S = \{(\text{sm}, \text{cir})\}$$

图 7.6 显示第二个训练例子后的变型空间。这时 H 仅含三个概念，它们满足上一个正例，但不满足这一个反例的全部概念。

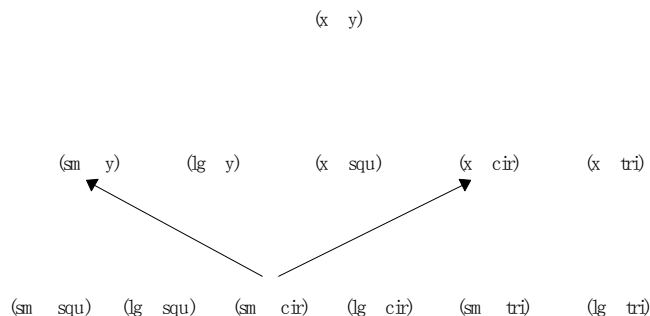


图 7.6 第二个训练例子后的变型空间

第三个训练例子是正例(lg cir)。这表示大圆是圆。这一步首先从 G 中去掉不满足此正例的概念(sm y)。再对 S 和该正例作泛化，得到

$$G=\{(x\ cir)\}$$

$$S=\{(x\ cir)\}$$

这时算法结束，输出概念($x\ cir$)。

对这个算法再补充说明几点。

(1) 对集合 G 和 S 的理解。对于符合所求概念的新例子，集合 S 是其充分条件的集合，集合 G 是其必要条件的集合。例如在第一个训练例子后，(sm cir) 是充分条件，即小圆一定满足所求概念。当时程序还不知道大圆是否满足。又如在第二个训练例子后，($x\ cir$) 和 (sm y) 是必要条件，即满足所求概念的例子或者是圆，或者是小的。算法结束时， $G=S$ ，即满足了充要条件。

(2) 学习正例时，对 S 进行泛化，这往往扩大 S 。学习反例时，对 G 进行特化，这往往扩大 G 。 G 和 S 的规模过大会给算法的实用造成困难。算法是在训练例子引导下，对规则空间进行宽度优先搜索。对大的规则空间，算法慢得无法接受。

7.4.2 两种改进算法

基本的变型空间学习方法很难实用，人们提出了一些改进的方法。其中两种改进的算法仅采用正例学习，它们类似于上述的修改 S 过程。

第一种是冲突匹配算法 (Hayes-Roth 和 McDormott)。它用于学习“参数化结构表示”所表达的概念。在上述的修改 S 过程中，总是对 S 作尽量少的泛化，以便覆盖新的正例。如果描述形式为谓词表达式，则这个过程相当于寻找最大的公共子表达式，这只需要去掉最少的合取条件。例如， S 集合为

$$S=\{BLOCK(x) \wedge BLOCK(y) \wedge SQUARE(y) \wedge RECTANGLE(x) \wedge ONTOP(x, y)\}$$

这表示积木 x 为矩形，积木 y 为正方形，且 x 在 y 上。若下一个训练正例 I_1 为

$$I_1=\{BLOCK(w) \wedge BLOCK(v) \wedge SQUARE(w) \wedge RECTANGLE(v) \wedge ONTOP(w, v)\}$$

这表示积木 w 为正方形，积木 v 为矩形，且 w 在 v 上。

经过修改 S 过程，将产生下列公共子集

$$S'=\{S_1, S_2\}$$

其中

$$S_1=BLOCK(a) \wedge BLOCK(b) \wedge SQUARE(a) \wedge RECTANGLE(b)$$

$$S_2=BLOCK(c) \wedge BLOCK(d) \wedge ONTOP(c, d)$$

S_1 相当于假设 ONTOP 这个位置关系与所求概念无关。 S_2 相当于假设积木形状与所求概念无关。应当注意，当 x 对应 w ，且 y 对应 v 时， S 与 I_1 的位置关系匹配，但形状特征不匹配。

反之,当 x 对应 v 而 y 对应 w 时, S 与 I_i 的形状特征匹配,但位置关系不匹配。这种现象是在匹配中的冲突。为了解决冲突,在 S' 中用两个元素分别考虑这两个方面。

第二种方法是最大的合一泛化。这个算法用于寻找谓词表达式的最大的合一泛化。它类似于冲突匹配算法,但是它使用的表示语言允许在匹配中多对一的参数联系。

变型空间方法有如下两个主要缺点:

(1) 抗干扰能力差。

所有数据驱动方法(包括变型空间方法)都难以处理有干扰的训练例子。由于算法得到的概念应满足每个训练例子的要求,所以一个错误例子会造成很大影响。有时错误例子使程序得到错误概念,有时得不到概念,这时 H 成为空集。

Mitchell(1978)提出的解决方法是保存多个 G 和 S 集合。例如, S_0 符合所有正例, S_i 符合除一个正例外其它的正例, S_2 等类似。如果 G_0 超过 S_0 , 则 H_0 为空集。这说明没有任何一个概念符合全部例子。于是程序去找 G_i 和 S_i , 以便得到 H_i 。如果 H_i 也空, 则找 H_2 。

(2) 学习析取概念。

变型空间方法不能发现析取的概念。有些概念是析取的。例如, PARENT 可能是父亲,也可能是母亲。这表示为 $\text{PARENT}(x) = \text{FATHER}(x) \vee \text{PARENT}(x) = \text{MOTHER}(x)$, 由于集合 G 和集合 S 的元素都是合取形式, 所以上述算法找不到析取概念。

第一种解决方法使用不含析取联结词的表示语言。它重复多次进行消除候选元素工作, 以找到覆盖全部例子的多个合取描述。

算法 7.2 学习析取概念算法

第一步: 集合 S 初始化为只含一个正例, 集合 G 初始化为没有描述。

第二步: 对每个反例, 进行修改 集合 G 。

第三步: 在 G 中选择一个描述 g , 把 g 作为解集合中的一个合取式。 g 不覆盖任一反例, 但会覆盖一部分正例。这时从正例集合中去掉比 g 特殊的所有正例(即 g 覆盖的那些正例)。

第四步: 对剩余的正例和全部反例, 重复一、二、三步, 直到所有正例都被覆盖。于是, 每次循环得到的 g 的析取就是所求概念。

这个析取不覆盖任一反例, 而且每一个 g 都不覆盖任一反例。该析取覆盖全部正例, 每一个 g 覆盖由它去掉的正例。注意, 由于没有修改 S 过程, 所以 g 不覆盖全部正例, 但 g 至少覆盖第一步那个正例, 所以 g 至少去掉这个正例。

第二个方法称为 AQ 算法[Michalski 1975]。这类似于上一种算法, 只是 AQ 算法在第一步用启发式方法选择一个正例, 要求这个正例是前几个 g 都没有覆盖过的。Larson 改进了 AQ 算法, 把它用于推广的谓词演算表示。

7.5 AQ 归纳学习算法

1969 年, Michalski 提出了 AQ 学习算法, 这是一种基于实例的学习方法。AQ 算法生成的选择假设的析取, 覆盖全部正例, 而不覆盖任何反例。它的基本算法如下:

算法 7.3 简单的 AQ 学习算法。

- (1) 集中注意一个实例(作为种子);
- (2) 生成该实例的一致性泛化式(称作 star);
- (3) 根据偏好标准, 从 star 选择最优的泛化式(假设)。如果需要, 特化该假设;
- (4) 如果该假设覆盖了全部实例, 则停止; 否则选择一个未被假设覆盖的实例, 转到 (2)。

Michalski 于 1978 年提出了 AQ11。它搜索规则空间, 反复应用消除候选元素, 得到尽可能

能一般的规则。AQ11 算法把学习鉴别规则问题转化为一系列学习单个概念问题。为了得到 C_i 类的规则，它把 C_i 类的例子作为正例，而把所有其它类的例子作为反例。由此找到覆盖全部正例而不包括任一反例的描述，以此作为 C_i 的规则。这样找到的鉴别规则可能在例子空间中未观察的区域内重迭。

为了寻找不重迭的分类规则集，AQ11 以 C_i 类的例子为正例，反例包括所有其它类型 $C_j (j \neq i)$ 的例子和已处理的各类型 $C_k (1 \leq k \leq i)$ 的正例区中的全部正例。于是， C_2 类只覆盖 C_1 类未覆盖的那部分， C_3 类覆盖的部分是在 C_2 和 C_1 类都不覆盖的部分。

AQ11 得到的鉴别规则相当于符合训练例子的最一般的描述的集合，即各类型的 G 集合 G_1, G_2 等。有时要使用符合训练例子的最特殊的描述的集合，即各类型的 S 集合 S_1, S_2 等。

Michalski 等采用 AQ11 程序学习 15 种黄豆病害的诊断规则。提供给程序 630 种患病黄豆植株的描述，每个描述是 35 个特征的特征向量。同时送入每个描述的专家诊断结论。选择例子程序从中选出 290 种样本植株作为训练例子。选择准则是使例子间相差较大。其余 340 种植株留作测试集合，用来检验得到的规则。

7.6 CLS 学习算法

CLS 学习算法是 1966 年由 Hunt 等提出的[Hunt 1966]。它是早期的决策树学习算法，后来的许多决策树学习算法都可以看作是 CLS 算法的改进与更新。

CLS 算法的主要思想是从一个空的决策树出发，通过添加新的判定结点来改善原来的决策树，直至该决策树能够正确地将训练实例分类为止。

算法 7.4 CLS 算法.

(1) 令决策树 T 的初始状态只含有一个树根 (X, Q) ，其中 X 是全体训练实例的集合， Q 是全体测试属性的集合；

(2) 若 T 的所有叶结点 (X', Q') 都有如下状态：或者第一个分量 X' 中的训练实例都属于同一各类，或者第二个分量 Q' 为空，则停止执行学习算法，学习的结果为 T ；

(3) 否则，选取一个不具有第 2 步所述状态的叶结点 (X', Q') ；

(4) 对于 Q' ，按照一定规则选取测试属性 b ，设 X' 被 b 的不同取值分为 m 个不相交的子集 X_i' ， $1 \leq i \leq m$ ，从 (X', Q') 伸出 m 个分叉，每个分叉代表 b 的一个不同取值，从而形成 m 个新的叶结点 $(X_i', Q' - \{b\})$ ， $1 \leq i \leq m$ ；

(5) 转 2。

从 CLS 算法的描述可以看出，决策树的构造过程也就是假设特化的过程，所以 CLS 算法可以看作是一个只带一个操作符的学习算法，此操作符可以表示为：通过添加一个新的判定条件（新的判定结点），特化当前假设。CLS 算法递归的调用这个操作符，作用在每个叶结点上，来构造决策树。

在算法 7.4 的步骤(2)中，如果训练实例集没有矛盾，即在没有所有属性的取值相同的两个实例属于不同的类，则如果第二个条件得到满足（即 Q' 为空），则第一个条件（即 X' 的所有训练实例都属于同一个类）也会得到满足，即停止条件只用二者中间的任何一个即可。但是对于可能存在的有矛盾的训练实例集，上述说法不一定成立。

在算法的步骤(4)中，应该满足 $m > 1$ 否则继续分类没有意义。但是，若 X 中有矛盾的训练实例，则难以保证 $m > 1$ 。

在算法的步骤(4)中，并未明确给出测试属性的选取标准，所以 CLS 有很大的改进空间。

7.6 ID3 学习算法

在算法 7.4 中并未给出如何选取测试属性 b ， $Hunt$ 曾经提出几种选择标准。但在决策树学习算法的各种算法当中，最为有影响的是 $Quinlan$ 于 1979 年提出的以信息熵的下降速度作为选取测试属性的标准的 ID3 算法[Quinlan 1979]。信息熵的下降也就是信息不确定性的下降。

7.6.1 信息论简介

1948 年 $Shannon$ 提出并发展了信息论，研究以数学的方法度量并研究信息。通过通信后对信源中各种符号出现的不确定程度的消除来度量信息量的大小。它提出了一系列概念：

(1) 自信息量。在收到 a_i 之前，收信者对信源发出 a_i 的不确定性定义为信息符号 a_i 的自信息量 $I(a_i)$ 。即 $I(a_i) = -\log p(a_i)$ ，其中 $p(a_i)$ 为信源发出 a_i 的概率。

(2) 信息熵。自信息量只能反映符号的不确定性，而信息熵可以用来度量整个信源 X 整体的不确定性，定义如下：

$$\begin{aligned} H(X) &= p(a_1)I(a_1) + p(a_2)I(a_2) + \dots \dots p(a_r)I(a_r) \\ &= -\sum_{i=1}^r p(a_i) \log p(a_i) \end{aligned} \quad (7.22)$$

其中 r 为信源 X 所有可能的符号数，即用信源每发一个符号所提供的平均自信息量来定义信息熵。

(3) 条件熵。如果信源 X 与随机变量 Y 不是相互独立的，收信者收到信息 Y 。那么，用条件熵 $H(X/Y)$ 来度量收信者在收到随机变量 Y 之后，对随机变量 X 仍然存在的不确定性。假设 X 对应信源符号 a_i ， Y 对应信源符号 b_j 。 $p(a_i/b_j)$ 为当 Y 为 b_j 时 X 为 a_i 的概率。则有：

$$H(X/Y) = -\sum_{i=1}^r \sum_{j=1}^s p(a_i b_j) \log p(a_i/b_j) \quad (7.23)$$

(4) 平均互信息量。用它来表示信号 Y 所能提供的关于 X 的信息量的大小，用 $I(X, Y)$ 表示：

$$I(X, Y) = H(X) - H(X/Y) \quad (7.24)$$

7.6.2 信息论在决策树学习中得以应用

在算法 7.4 中，我们在学习开始的时候只有一棵空的决策树，并不知道如何根据属性将实例进行分类，我们所要做的就是根据训练实例集构造决策树来预测如何根据属性对整个实例空间进行划分。设此时训练实例集为 X ，目的是将训练实例分为 n 类，设属于第 i 类的训练实例个数是 C_i ， X 中总的训练实例个数为 $|X|$ ，若记一个实例属于第 i 类的概率为 $P(C_i)$ 则：

$$P(C_i) = \frac{C_i}{|X|} \quad (7.25)$$

此时决策树对划分 C 的不确定程度为：

$$H(X, C) = -\sum P(C_i) \log P(C_i) \quad (7.26)$$

以后在无混淆的情况下将 $H(X, C)$ 简记为 $H(X)$ 。

决策树学习过程就是使得决策树对划分的不确定程度逐渐减小的过程。若选择测试属性 a

$$\begin{aligned} H(X/a) &= -\sum_i \sum_j p(C_i; a = a_j) \log p(C_i/a = a_j) \\ &= -\sum_i \sum_j p(a = a_j) p(C_i/a = a_j) \log p(C_i/a = a_j) \\ &= -\sum_j p(a = a_j) \sum_i p(C_i/a = a_j) \log p(C_i/a = a_j) \end{aligned} \quad (7.27)$$

进行测试，在得知 $a = a_j$ 的情况下属于第 i 类的实例个数为 C_{ij} 个。记

$$P(C_i; a = a_j) = \frac{C_{ij}}{|X|}, \text{ 即 } P(C_i; a = a_j) \text{ 为在测试属性 } a \text{ 的取值为 } a_j \text{ 时它属于第 } i \text{ 类的概率。}$$

此时决策树对分类的不确定程度就是训练实例集对属性 X 的条件熵。

$$H(X_j) = -\sum_i p(C_i/a = a_j) \log p(C_i/a = a_j) \quad (7.28)$$

又因为在选择测试属性 a 后伸出的每个 $a = a_j$ 叶结点 X_j 对于分类信息的信息熵为固有：

$$H(X/a) = \sum_j p(a = a_j) H(X_j) \quad (7.29)$$

属性 a 对于分类提供的信息量为 $I(X; a)$ ：

$$I(X; a) = H(X) - H(X/a) \quad (7.30)$$

式 (7.29) 的值越小则式 (7.30) 的值越大，说明选择测试属性 a 对于分类提供的信息越大，选择 a 之后对分类的不确定程度越小。Quinlan 的 ID3 算法就是选择使得 $I(X; a)$ 最大的属性作为测试属性，也即选择使得式 (7.29) 最小的属性 a 。

7.6.3 ID3 算法

在 ID3 算法中除去以信息论测度作为标准之外，还引入了增量式学习的技术。在 CLS 算法中，因为每次运行开始的时候算法要知道所有训练实例，当训练实例集过大的时候，实例无法立刻全部放入内存，会发生一些问题。Quinlan 在 ID3 算法中引入了窗口 (windows) 的方法进行增量式学习来解决这个问题。下面给出 ID3 算法。

算法 7.5 ID3 算法

- (1) 选出整个训练实例集 X 的规模为 W 的随机子集 X_1 (W 称为窗口规模，子集称为窗口)；
- (2) 以使得 (7.29) 式的值最小为标准，选取每次的测试属性形成当前窗口的决策树；
- (3) 顺序扫描所有训练实例，找出当前的决策树的例外，如果没有例外则训练结束；

(4)组合当前窗口的一些训练实例与某些在(3)中找到的例外形成新的窗口，转(2)。

为了在步骤(4)建立新的窗口，Quinlan 试验了两种不同的策略：一个策略是保留窗口的所有实例，并添加从步骤(3)中获得的用户指定数目的例外，这将大大扩充窗口；第二个策略是相当于当前决策树的每一个叶结点保留一个训练实例，其余实例则从窗口中删除，并用例外进行替换。实验证明两种方法都工作得很好，但是如果概念复杂到不能发现固定规模 W 的任意窗口的时候，第二种方法可能不收敛。

7.6.4 ID3 算法应用举例

表 7.2 给出了一个可能带有噪音的数据集合。它有四个属性，Outlook、Temperature、Humidity、Windy。它被分为两类，P 与 N，分别为正例与反例。所要做的就是构造决策树将数据进行分类。

因为初始时刻属于 P 类和 N 类的实例个数均为 12 个，所以初始时刻的熵值为：

$$H(X) = -\frac{12}{24} \log \frac{12}{24} - \frac{12}{24} \log \frac{12}{24} = 1$$

如果选取 Outlook 属性作为测试属性则有，根据公式 (2.8)，此时的条件熵为：

$$\begin{aligned} H(X / \text{Outlook}) &= \frac{9}{24} \left(-\frac{4}{9} \log \frac{4}{9} - \frac{5}{9} \log \frac{5}{9} \right) + \frac{8}{24} \left(-\frac{1}{8} \log \frac{1}{8} - \frac{7}{8} \log \frac{7}{8} \right) \\ &+ \frac{7}{24} \left(-\frac{1}{7} \log \frac{1}{7} - \frac{6}{7} \log \frac{6}{7} \right) = 0.5528 \end{aligned}$$

如果选取 Temperature 属性作为测试属性则有：

$$\begin{aligned} H(X / \text{Temp}) &= \frac{8}{24} \left(-\frac{4}{8} \log \frac{4}{8} - \frac{4}{8} \log \frac{4}{8} \right) + \frac{11}{24} \left(-\frac{4}{11} \log \frac{4}{11} - \frac{7}{11} \log \frac{7}{11} \right) \\ &+ \frac{5}{24} \left(-\frac{4}{5} \log \frac{4}{5} - \frac{1}{5} \log \frac{1}{5} \right) = 0.6739 \end{aligned}$$

如果选取 Humidity 属性作为测试属性则：

$$H(X / \text{Humid}) = \frac{12}{24} \left(-\frac{4}{12} \log \frac{4}{12} - \frac{8}{12} \log \frac{8}{12} \right) + \frac{12}{24} \left(-\frac{4}{12} \log \frac{4}{12} - \frac{8}{12} \log \frac{8}{12} \right) = 0.9183$$

如果选取 Windy 属性作为测试属性则有：

$$\begin{aligned} H(X / \text{Windy}) &= \frac{8}{24} \left(-\frac{4}{8} \log \frac{4}{8} - \frac{4}{8} \log \frac{4}{8} \right) + \frac{6}{24} \left(-\frac{3}{6} \log \frac{3}{6} - \frac{3}{6} \log \frac{3}{6} \right) \\ &+ \frac{10}{24} \left(-\frac{5}{10} \log \frac{5}{10} - \frac{5}{10} \log \frac{5}{10} \right) = 1 \end{aligned}$$

表 7.2 样本数据集合

属性	Outlook	Temperature	Humidity	Windy	类
1	Overcast	Hot	High	Not	N
2	Overcast	Hot	High	Very	N
3	Overcast	Hot	High	Medium	N
4	Sunny	Hot	High	Not	P
5	Sunny	Hot	High	Medium	P
6	Rain	Mild	High	Not	N
7	Rain	Mild	High	Medium	N
8	Rain	Hot	Normal	Not	P
9	Rain	Cool	Normal	Medium	N
10	Rain	Hot	Normal	Very	N
11	Sunny	Cool	Normal	Very	P
12	Sunny	Cool	Normal	Medium	P
13	Overcast	Mild	High	Not	N
14	Overcast	Mild	High	Medium	N
15	Overcast	Cool	Normal	Not	P
16	Overcast	Cool	Normal	Medium	P
17	Rain	Mild	Normal	Not	N
18	Rain	Mild	Normal	Medium	N
19	Overcast	Mild	Normal	Medium	P
20	Overcast	Mild	Normal	Very	P
21	Sunny	Mild	High	Very	P
22	Sunny	Mild	High	Medium	P
23	Sunny	Hot	Normal	Not	P
24	Rain	Mild	High	Very	N

可以看出 $H(X/\text{Outlook})$ 最小，即有关 Outlook 的信息对于分类有最大的帮助，提供最大的信息量，即 $I(X; \text{Outlook})$ 最大。所以应该选择 Outlook 属性作为测试属性。并且也可以看出 $H(X) = H(X/\text{Windy})$ ，即 $I(X; \text{Windy}) = 0$ ，有关 Windy 的信息不能提供任何有关分类的信息。选择 Outlook 作为测试属性之后将训练实例集分为三个子集，生成三个叶结点，对每个叶结点依次利用上面过程则生成图 2.4 的决策树。

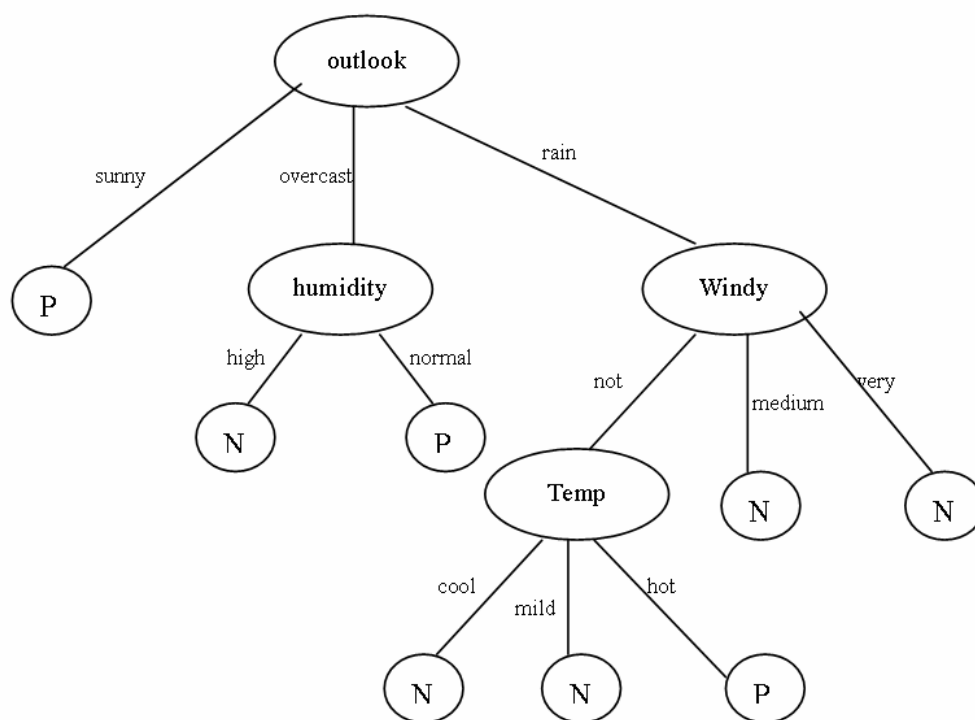


图 7.7 表 7.2 所训练生成的决策树

ID3 算法有着广泛的应用，其中比较著名的是 C4.5 系统。C4.5 的新功能是它能够将决策树转换为等价的规则表示，并且 C4.5 解决了连续取值的数据学习问题。

7.6.5 连续性属性离散化

决策树主要是用来学习离散性变量作为的属性类型的学习方法，连续性变量必须被离散化才能够被学习。然而在一些算法当中（例如 C4.5），使得被离散化的连续性属性相对于离散变量的属性来说更容易被选取。在这些算法当中，对于某个连续属性，首先将储存的训练实例中的不同的值进行排序，然后选取每一对相邻的值的中点作为离散化时区分不同属性值的标准。因为这些离散化后的属性值只有一个实例为代表，则这些离散化后的连续属性将被优先选取。

Dougherty 采取了一种基于信息熵的过程对连续型的属性离散化。他们在进行决策树生成之前对连续性变量全局的离散化，而不只是象在 C4.5 中那样只是基于某个结点的实例来进行局部的离散化。因为局部的数据过于少而更容易受到数据当中噪音的干扰。这种方法基于信息熵不断递归地对连续型属性的值进行二分来产生由多个属性值的离散型属性，并且采用基于 MDL 标准停止标准。他们发现这种方法用于 C4.5 并不降低整个分类的正确性，反而有时会提高它的正确性，并且它能够减小 C4.5 中树的大小。

Auer 提出了一个用局部的方法离散化连续型变量的方法，他的 T2 算法也是将连续变量分为多个离散变量而不只是二分连续变量，它并不是项上面那样递归地对连续型变量二分，而是进行彻底的搜索以找出一组 m 个间隔来使得在训练实例集上错误率的最小。 m 的缺省值是 $C+1$ ，这里 C 是所要划分的类的数目。所以 T2 的复杂度与 $C^6 f^2$ 成正比， f 是属性个数。

在 C4.5 Release8 (R8) 中 Quinlan 提出了一种局部的、基于 MDL 的方法来对离散化后的连续型属性的如果取值过多进行惩罚 (penalize)。

实验数据表明, C4.5 R8 的方法在大数据集合当中进行离散化效果比较好, 而 Dougherty 的全局离散化的方法在小数据集合当中效果比较好。T2 在数据被分为较少的类的时候工作较好。

7.7 基于偏置变换的决策树学习算法 BSDT

构造好的决策树的关键在于选择好的属性。一般情况下, 在众多能够拟合给定训练例子的决策树中, 树越小则树的预测能力越强。要构造尽可能小的决策树, 关键在于选择恰当的属性。由于构造最小树问题是个 NP 完全问题, 因此大量的研究只能采取用启发式策略选择好的属性。属性选择依赖于各种对例子子集的杂质度量方法。杂质度量方法包括信息增益、信息增益比、Gini-index、距离度量、J-度量、G 统计、 χ^2 统计、 P^0 零假设概率、证据权值、最小描述长(MDL)、正交法度量、相关度和 Relief 等。不同的度量有不同的效果, 因此还出现了单元决策树与多元决策树的分别。对这些度量方法的比较进行了较为深入的研究, 结论却是莫衷一是的, 没有一种算法在解决属性选择、数据噪声、数据递增、预剪枝与后剪枝及剪枝停机标准等问题时占绝对优势。在一定程度上, 经验的和直觉的东西代替了严密完备的理论证明。

上述问题实际上就是关于决策树学习中的偏置问题。偏置在概念学习中具有重要作用。Utgoff 指出没有偏置就没有归纳学习。所谓偏置, 是指概念学习中除了原始训练例子集以外影响假设选择的所有因素。这些因素包括: 描述假设的语言; 程序考虑假设的空间; 按什么顺序假设的过程; 承认定义的准则等。偏置有两个特点: 一是强偏置将概念学习集中于相对少量的假设。反之, 弱偏置允许概念学习考虑相对大量的假设; 另一个特点则是正确偏置允许概念学习选择目标概念, 不正确的偏置则不能选择目标概念。当偏置很强且正确时, 概念学习能立即选择可用的目标概念, 当偏置很弱且不正确时, 概念学习的任务是十分困难的。

偏置可分为表示偏置与过程偏置两类。由于 ID3 算法家族缺少对背景知识的支持, 是一个相对较弱的偏置支持下的归纳学习方法, 我们通过对表示偏置及过程偏置的变换, 强化决策树学习的偏置[199]。

7.7.1 偏置的形式化

在实现偏置变换之前, 我们有必要对文中涉及到的各个基本概念予以形式化的描述。

定义 7.1 S 是定义在 $\langle A, C, F, T, L \rangle$ 上的一个搜索空间。其中: 属性向量 $A = \{a_1, \dots, a_m\}$ 具有有限或无限个元素; 类别向量 $C = \{c_1, \dots, c_k\}$ 有有限个元素, 对于给定的 A 和 C , F 是所有概念的集合, T 是所有元组数为 n 的训练例子的集合; 其中 L 表示一个学习算法簇。

定义 7.2 一个学习算法 l 定义了一个从 T 到 F 的映射, 即:

$$t \{t \in T\} \xrightarrow{l(t, f)} f \{f \in F\}, l \in L \quad (7.31)$$

定义 7.3 $D_{A \times C}$ 是在 $A \times C$ 上的一个概率分布; t 是定义在 $A \times C$ 上并服从 $D_{A \times C}$ 的 n 个元组。设 D_C 是在 C 上的一个概率分布, 且属性集的同伦性 $IA(A_1, A_2)$ 是指在给定概念 f 及 D_C 的情况下将一随机属性赋予同一类的概率。即:

$$IA(A_1, A_2) = P_{D_A}(l(T_0(A_1, f)) = l(T_0(A_2, f))), l \in L \cap T_0 \subseteq T \cap A_1, A_2 \subseteq A \cap f \in F$$

定义 7.4 假设目标概念为 f_g , 偏置的正确性 $CorrB$ 可定义为:

$$CorrB = P_{D_A}(f_g(a) = f(a)), f \in F \cap a \in A \quad (7.32)$$

定义 7.5 设 $|S|$ 表示 S 中元素个数, 则可定义偏置的强度 $StrB$ 为:

$$StrB = \frac{1}{|S|} \quad (7.33)$$

定义 7.6 设 $State_0(S) = \langle A_0, C, f, T, \triangleright \rangle$ 与 $State_1(S) = \langle A_1, C, f, T, \triangleright \rangle$ 为搜索空间的两个状态, 表示偏置变换 BS^R 定义为:

$$State_0(S) \xrightarrow{BS^R} State_1(S)$$

定义 7.7 设 D_A 是在 A 上的一个概率分布, 且学习算法的同一性 $IL(l_1, l_2)$ 是指在给定概念 f 及 D_A 的情况下 l_1 及 l_2 将一随机训练例子 t 赋予同一类的概率。即:

$$IL(l_1, l_2) = P_{D_A}(l_1(t, f) = l_2(t, f)), t \in T \cap l_1, l_2 \in L \cap f \in F \quad (7.34)$$

定义 7.81 的预测准确性 PA 定义为:

$$PA(l) = P_{D_{A \times C}}(f_{l(t)}(a) = c), f \in F \cap t \in T \cap c \in C \quad (7.35)$$

定义 7.9 设 $State_0(S) = \langle A, C, f, T, l_0 \rangle$ 与 $State_1(S) = \langle A, C, f, T, l_1 \rangle$ 为搜索空间的两个状态, 过程偏置变换 BS^P 定义为:

$$State_0(S) \xrightarrow{BS^P} State_1(S)$$

定理 7.1 设 l_1 及 l_2 为学习算法, 当 $PA(l_1) \geq PA(l_2)$ 成立时, 选择 l_1 具有更正确的偏置。

证明 当 $PA(l_1) \geq PA(l_2)$ 成立时, 说明由 l_1 生成的分类器能将更多的例子准确地分类。即:

$$P_{D_{A \times C}}(f_{l_1(t)}(a) = c) \geq P_{D_{A \times C}}(f_{l_2(t)}(a) = c) \quad (7.36)$$

我们将偏置正确性概念投射到 $D_{A \times C}$ 分布上以后, 得到:

$$CorrB = P_{D_{A \times C}}(f_g(a) = c)$$

由于目标概念在很大程度上是由学习算法生成的分类器来具体体现的, 因而偏置正确性概念可以改写为:

$$CorrB = P_{D_{A \times C}}(f_{l(t)}(a) = c)$$

将算法 l_1 及 l_2 代入上式并结合(7.36)式可以得到:

$$CorrB_1 = P_{D_{A \times C}}(f_{l_1(t)}(a) = c) \geq CorrB_2 = P_{D_{A \times C}}(f_{l_2(t)}(a) = c)$$

即: 选择 l_1 具有更正确的偏置。结论得证。

7.7.2 表示偏置变换

决策树学习算法是实际有效的, 但由于缺乏对背景知识的支持, 它不能处理各种类型的泛化关系, 而对于基于谓词逻辑的归纳算法 (如 AQ11、INDUCE 等), 这种功能是最基本且

与学习过程本身密不可分的。缺乏背景知识的后果之一就是使决策树生成复杂化且不易于为领域专家所理解。

很多系统都试图解决这个问题。例如：Cendrowska 的 PRISM 算法, Gaines 的 INDUCT 算法, 以及后来 Quinlan, Lavrac 等援引其它技术将决策树修改得更易懂、更精确, 但它们都仅仅将注意力集中于包含在历史数据中的信息, 力图从中挖掘出更多一点的可用资源。

我们提出了一种使基于表示偏置变换的、能够处理不同泛化关系的学习算法的预处理算法。该方法首先对原始训练例子集进行预处理, 调用泛化算法 CGAOI, 使原始训练例子集达到指定的概念层次; 然后再对原始训练例子集进行预处理。

为了实现提出的方法, 我们首先引入了概念层次的概念。

定义 7.10 一个概念层次是从一组较低级概念向高级概念的映射序列。一棵概念层次树则是将这些映射序列以树的形式组织起来。

定义 7.11 一个原始训练集 E^0 对应于一个概念层次森林 $F = \{F_1, \dots, F_m\}$ 。其中, F_i 是与第 i 个属性 A_i 相对应的一棵概念层次树集合, $1 \leq i \leq m$ 。 $F_i = \{T_1, \dots, T_\tau\}$, 其中, T_j 是属性 A_i 的第 j 棵概念层次树, $1 \leq j \leq \tau$ 。

定义 7.12 当属性 A_i 不存在概念层次树时, $F_i = \text{NULL}$ 。

定义 7.13 一个概念层次库 D 用于保存历史概念层次树。

在原始训练集的所有属性中, 许多属性都存在各自固有的概念层次。例如, 一个商品的产地在北京, 我们就可以说是在中国, 进而也可以说产自亚洲。这就是一个 $\{\text{Product Place : Beijing, China, Asia}\}$ 的概念层次。

概念层次用于表达控制泛化及特化过程所必需的背景知识。通过将不同层次的概念组织成一个树形分类, 概念空间可以从泛化到特化的顺序偏序地表示。最泛化的概念就是内涵为空的描述, 可用关键字“ANY”来表示; 而最特化的概念就是树形分类中的叶子。利用一个概念层次, 我们就可以将所发现的规则表述为简洁明了、更易懂、更泛化及逻辑性更强的形式。

7.7.3 算法描述

1. 分类制导的面向属性算法 CGAOI

分类制导的面向属性归纳方法 CGAOI 能够将原始关系泛化到指定的概念层次。我们提出了由分类制导的面向属性归纳方法 CGAOI。该方法是一种有教师学习方法。它作为分类学习任务的先导, 对原始训练例子集进行预处理, 并柔性地输出指定层次的泛化关系。

该算法在原始训练例子集的分类特征的制导下, 在实现面向属性归纳的基本泛化操作如: 属性去除、概念提升、属性阈值控制及频数传播等操作的同时, 实现泛化一致性检查、数据噪声去除、概念层次自动生成等操作。

算法 7.6 泛化算法 CGAOI。

输入: 原始训练例子集 E^0 , 属性集 A^0 , 当前属性 A ; 概念层次树 T , 指定概念层次 L ; 指定属性控制阈值 I ; 当前概念层次, 当前属性控制阈值 I^0 。

输出: 指定概念层次训练例子集 E , 属性集 A , 频数 C^r 。

- (1) 调用算法 GCCC 进行泛化一致性检查与噪声去除, 返回 Ret1;
- (2) 如果泛化一致性检查失败, 则 return -1;
- (3) 在 I 及 L 控制下作概念提升操作;
- (4) 属性去除及频数传播操作;

(5) 返回 0。

定理 7.2 令 $BS^R = \text{CGAOI}$ 算法, 则通过 CGAOI 操作进行的表示偏置变换得到的偏置是更强且正确的。

证明 设变换前后的属性集、偏置正确性、偏置强度分别为 A_0 、 A_1 ; CorrB_0 、 CorrB_1 ; StrB_0 、 StrB_1 。

① 由于 CGAOI 算法并没有改变目标概念以及在 C 的概率分布, 因此 CGAOI 操作得到的偏置并不改变原偏置的正确性;

② 由于 CGAOI 算法使得 $A_1 \leq A_0$, 从而有 $|S_1| = |A_1 \times C \times F \times T \times L| \leq |S_0| = |A_0 \times C \times F \times T \times L|$, 由偏置强度定义可以推出: $\text{StrB}_1 \geq \text{StrB}_0$ 。

从①、②可以证得: 通过 CGAOI 算法进行的表示偏置变换得到的偏置是更强且正确的。

2. 预处理算法 PPD

在该算法中, 我们以数据库中特定的属性值作为泛化过程及特化过程的分界, 当属性控制阈值大于当前属性值 (或层次级别低于当前级别) 时, 调用泛化过程; 反之则调用特化过程将原始训练例子集归约到相应的概念层次中。

算法 7.7 预处理算法 PPD。

输入: 原始训练例子集 E^0 ; 属性集 A^0 ; 概念层次森林 F ; 概念层次库 D ; 指定概念层次 L ; 指定属性控制阈值 Y ; 当前概念层次; 当前属性控制阈值 Y^0 ;

输出: 指定概念层次训练例子集 E ; 属性集 A ;

(1) 对属性集合 A^0 中的每个属性 A_i 做:

(2) A_i 的概念层次树 F_i 是否为空;

① 空则调用算法 AGCH 自动生成 A_i 的概念层次树 F_i , 返回 Ret1;

② Ret1=-1, 则转到步骤 1; /* A_i 无概念层次树 */

(3) 如果 $(Y = Y^0 \cap L = L^0)$, 则转到步骤 (1);

(4) 如果 $(Y < Y^0 \cup L > L^0)$, 则调用泛化算法 CGAOI, 返回 Ret2;

① Ret2=-1, 则转到步骤 (1); /* 失败, 则放弃泛化 */

② Ret2=0, 则转到步骤 (7); /* 泛化成功 */

(5) 不满足上述条件, 则转到步骤 1;

(6) 调用算法 MEA 进行训练例子集及属性集维护;

(7) 调用算法 STCH 将 A_i 的概念层次树 T 存入概念层次库 D 中;

(8) 转步骤 (1)。

7.7.4 过程偏置变换

各种决策树学习算法各有所长, 博采众长正是我们所提出的基于偏置变换的决策树学习算法的基本点。同时, 学习算法及其学习任务与训练集的大小、维数、所属领域等属性有着复杂的关系, 不能用简单的控制分支语句来实现对最优算法的选择。因此, 我们引入了两级式基于偏置变换的决策树学习算法的概念 (见图 7.8)。

本算法是基于两级式及多策略的思想进行设计的。两级式学习的着眼点在于: 第一级的基于范例推理用于解决从众多评价指标不一、适应领域不一、训练例子规模不一的决策树学习算法中找到一个最适合解决原始训练例子集的学习算法; 第二级的学习任务用于分类器的构造, 即利用选出的决策树学习算法从一组无次序、无规则的事例中推理出决策树表示形式的分类规则。

基于范例推理是由目标范例的提示而获得记忆中的源范例，并由源范例来指导目标范例求解的一种策略。这里，目标范例是由各种决策树算法运行经典范例库而生成的。对一给定原始训练例子集，我们首先抽取它的检索信息元组 θ 作为检索信息到范例库中进行检索。检索过程的相似度符合定义 7.25 中给出的择优指标 ζ 。

多策略学习体现在本算法不局限于一个学习算法。同时还为引入新算法和经典例子集提供了机制。这种机制保证了新引入算法与原注册算法、新引入经典范例集与原注册经典范例集间的无缝连接。这种机制是由人机交互纳入接口及经典范例库维护算法实现的。

定义 7.14 偏置系数 $C^b = [C^b_1, \dots, C^b_m]^T$ 表示原始训练集的各个属性与待学习任务的相关程度；其中： m 是原始训练例子集的属性个数。

定义 7.15 代价系数 $C^c = [C^c_1, \dots, C^c_m]^T$ 表示为得到原始训练集的各个属性所付出的代价；其中： m 是原始训练例子集的属性个数

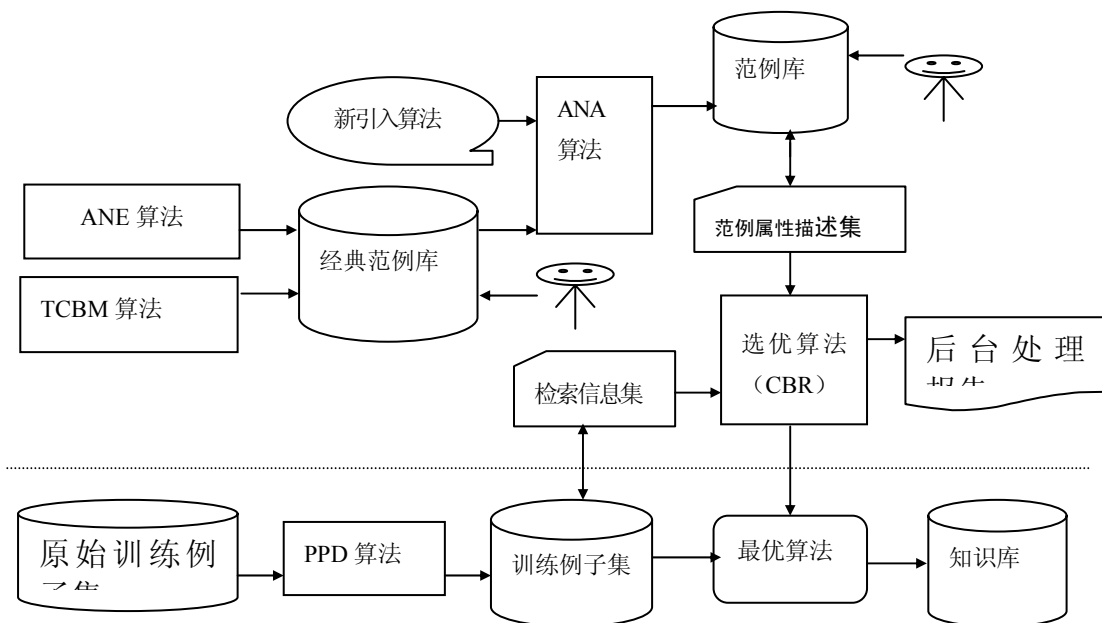


图 7.8 BSDT 算法结构

定义 7.16 原始训练例子集 E 是非 BSDT 算法构造的面向目标学习任务的训练例子集。

定义 7.17 一个算法 α 是一个在 BSDT 中注册的且已被实验证实可行的决策树学习算法。一个算法的名字是该算法的唯一标志。一个算法集 A 是在 BSDT 算法中注册的所有算法 α 的集合。

定义 7.18 算法索引表 AI 是在 BSDT 算法中注册的所有算法 α 的名字的集合。算法集 A 与算法索引表 AP 存在着——对应的关系。

定义 7.19 范例属性描述集 Θ 是一个定义在经典范例库上的六元组 $\langle \pi, S^e, S^d, S^t, \eta, N^e \rangle$ 。其中： π 表示由算法 δ 处理的经典训练例子集 E^t 所属的域名； S^e 表示 E^t 的大小； S^d 表示 E^t 的维数； S^t 表示符合 E^t 且由算法 δ 生成决策树所需的时间； η 表示该决策树的错分率； N^e 表示算法 δ 的名字。这里 N^e 作为范例库的类别标志。

定义 7.20 检索信息元组 θ 是一个定义在原始训练例子集上的三元组 $\langle a_1, a_2, a_3 \rangle$ 。其中， a_1 表示给定的原始训练例子集 E 所属的域名； a_2 表示 E 的大小； a_3 表示 E 的维数。

定义 7.21 范例库 CE 是由 BSDT 算法生成的范例的集合。它的六个属性由范例属性描述集 Θ 描述；类别 $N^e \in$ 算法索引表 AP ；例子 $ce \in CE$ 由算法 ANA 生成。

定义 7.22 域名 $\pi \in$ 域名集合 Π 。在 BSDT 中， $\Pi = \{1\text{-农业}, 2\text{-工业}, 3\text{-商业}, 4\text{-教育}, 5\text{-电子}, 6\text{-物理}, 7\text{-化学}, 8\text{-数学}, 9\text{-医药}, 10\text{-其它}\}$ 。

定义 7.23 经典范例库 $TEB =$ 例子索引表 $EI \cup$ 经典范例表 TET 。例子索引表 EI 是在 BSDT 算法中注册的所有域名 π 的集合；经典范例表 TET 是一个以域名 π 命名的从某一领域得到训练例子集合。

定义 7.24 一个选优指标 ζ 是由范例属性描述集 Θ 与检索信息元组 θ 共同决定的范例检索标准。它由下式确定：

$$\zeta = (a_1 = \pi) \wedge (|a_2 - S^e| < \lambda_1) \wedge (|a_3 - S^d| < \lambda_2) \wedge (S^t \cdot \eta < \lambda_3)$$

其中， $\lambda_1, \lambda_2, \lambda_3$ 分别是元组阈值，维数阈值及树控阈值，在算法运行过程中可调。在 BSDT 中， $\lambda_1, \lambda_2, \lambda_3$ 的缺省值分别为（其中， T_{in} , E_s , E_n 分别为经典范例库中的元组间隔、错分率及生成时间的期望值）：

$$\lambda_1 = 0.1 \times T_{in}, \lambda_2 = 2, \lambda_3 = \frac{E_s \times E_\eta}{\sqrt{2}}$$

定义 7.25 一个算法是最优算法 Γ 当且仅当它是满足选优指标 ζ 且使 λ_3 达到最小的算法。

定义 7.26 后台算法处理报告 REP 是一个文本文件 ErrRep.txt。当 BSDT 在指定路径找不到最优算法 Γ 时，就会将出错信息记录下来，通知管理人员处理。

定义 7.27 在 BSDT 算法中注册表示下列过程：

- (1) 将一个算法的名字 N^a 存入算法索引表 AI，将算法本身存放在 BSDT 指定路径下；
- (2) 将一个例子集的名字 N^{es} 存入例子索引表 EI，将经典范例表 TET 存放在经典范例库中。

7.7.5 基于偏置变换的决策树学习算法 BSDT

BSDT 算法首先调用选优算法 SM 求出最优算法 Γ 。然后就到系统目录及当前工作目录下寻找该算法。如果没有找到就应该从范例库及算法索引表中去掉它，以免引起下一次寻找失败。

选优算法 SM 实际上是一个运用基于范例推理的过程。对一给定的原始训练集，首先抽取它的检索信息元组 θ ，再到范例库中查找出满足选优指标 ζ 的最优算法。当范例库空时或 SM 算法失败时，令 GSD 作为最优算法的默认选择。

由于在算法运行中应减少不必要的干扰，BSDT 日志在此可作为管理员对系统进行干预的一个手段。

算法 7.8 基于偏置变换的决策树学习算法 BSDT。

输入：原始训练集；

输出：目标知识库；

- (1) 调用预处理算法 PPD；
- (2) 调用选优算法 SM 求出最优算法 Γ ；
- (3) 在指定路径上搜索算法 Γ ；
- (4) 如果算法存在：
 - ① 调用 Γ ；
 - ② 生成决策树 and/or 产生式规则集；
 - ③ 将学习结果处理后存入目标知识库。
- (5) 算法不存在：
 - ① 令 $p=3$ ，调用范例库维护算法 CBM 从范例库及相应数据结构中删去 Γ ；
 - ② 转 (2)；
- (6) 填写 BSDT 日志；

7.7.6 经典范例库维护算法 TCBM

TCBM 算法实现对经典范例库的各种操作，如：增加新的例子集、例子集版本升级、删除例子集以及向例子库的指定例子集中增加、删除例子等。只有当有新例子及例子集的增删及修改已有例子集时才能激活该算法。由于采取了逐级激活的方式，TCBM 算法在修改了经典范例库后，会自动调用在算法索引表中的各算法重新执行以构造与经典范例库内容一致的范例库，而范例库及相关数据结构的修改以会直接影响最优算法的选取。

算法 7.9 经典例子库维护算法 TCBM。

输入：原始例子库、例子索引表，例子集 S_0 ，域名 π ，维护操作算子 ρ

输出：修改后例子库、例子索引表

- (1) IF $\rho < 1$ and $\rho > 5$ THEN 返回。
- (2) IF $\rho = 1$, THEN: /* 增加新例子集 */
 调用 ANE 算法增加新例子;
- (3) IF $\rho = 2$, THEN: /* 经典范例库版本升级 */
 - ① 检查例子索引表，域名 π 是否存在;
 - ② 存在：
 - a 从例子库删除相应表;
 - b 删除例子索引表中相关项;
 - ③ 调用 ANE 算法引入新例子集;
- (4) IF $\rho = 3$, THEN: /* 例子集删除 */
 - ① 检查例子索引表，域名 π 是否存在，否则返回;
 - ② 从库中删除相应表;
 - ③ 从例子索引表中删除域名 π ;
- (5) IF $\rho = 4$, THEN: /* 增加例子库例子 */
 - ① 若域名 π 不在索引表中，则调用 ANE 算法增加新域;
 - ② 进行记录值的适当性检测，不适当的记录值被舍弃；将训练集 S_0 依次存入相应表中;
- (6) IF $\rho = 5$, THEN: /* 删去例子库例子 */
 - ① 若域名 π 不在索引表中，则返回;
 - ② 删去表中例子;
- (7) Extract(S_0 , π); /* 为经典例子库生成例子 */
- (8) 存储例子;

7.7.7 偏置特征抽取算法

偏置，也就是候选算法的特征能够帮助我们利用选优算法自动选取最优偏置。这些特征按照定义 7.19 的形式进行组织。当经典例子库或算法索引表发生变化时，该过程即被激活。

算法 7.10 偏置特征抽取算法

输入：经典例子库 CEB，注册算法 RA，例子集 ES，例子索引表 EI，算法索引表 AI，范例库 CB，操作算子 OP

输出：修改后的 CB

- (1) If op=1 then /* CB 是空的 */
 { If EI = NULL || AI = NULL return;
- (2) For $i=1$ to maxRA
- (3) For $j=1$ to maxEI
- (4) For $k=1$ to Interval do
 { 用算法 AI(i) 在例子集 EI(j) 上生成一个分类器，并根据定义 7.19 生成一个数据项;
 将数据项存入 CB; }
- (5) Else if op=2 then /* 添加一个新算法 */

```

(6) { For  $i=1$  to maxEI
(7)   For  $j=1$  to Interval do
      { 用算法RA在例子集EI( $j$ )上生成一个分类器, 并根据定义7.19生成一个数据项;
        将数据项存入CB; }
(8) Else   if op=3 then           /* 添加一个新的例子集 */
(9) { For  $i=1$  to maxRA
(10)   For  $j=1$  to Interval do
      { 用算法AI( $i$ )在例子集ES上生成一个分类器, 并根据定义7.19生成一个数据项;
        将数据项存入CB; }
(11) Else if op=4 then           /* 删除一个老的注册算法 */
      从CB中删除RA的相关项;
(12) Else if op=5 then          /* 删除一个老的注册例子集 */
      从CB中删除ES的相关项;
(13) Else return failed;
      根据CEB的大小调整Interval;

```

7.7.8 改进的决策树生成算法 GSD

在构造ASF函数时, 我们用到了代价系数 C^c 及偏置系数 C^b , 并为它们建立了缺省设置。下面给出了属性选择函数ASF的构造方法。

定义7.28 $E^0 = \{e_1^0, \dots, e_v^0\}$ 是原始训练例子的集合, $A^0 = \{A_1^0, \dots, A_m^0\}$ 是 E^0 的属性集。其中, v 是最大原始训练例子数, m 是 E^0 的最大属性数; $A_i = \{V_{i1}, \dots, V_{iv}\}$ 表示属性 i 的 v 个相互区别的取值。

定义7.29 $E = \{e_1, \dots, e_n\}$, $A^T = A \cup A^X = \{A_1, \dots, A_m\} \cup A^X = \{A_1, \dots, A_m, A_{m+1}\}$, 是通过引入背景知识及用CGAOI进行预处理后得到的训练例子集与属性集。其中, $1 \leq n \leq v$; $2 \leq m \leq m$ 。

定义7.30 $A^T = C^T = [C_1^T, \dots, C_n^T]^T$ 是在CGAOI算法中得到的传播频数。

定义7.31 $C = \{C_1, \dots, C_k\}$ 是原始训练例子集 E 中 k 个可能的类别。 P_i 表示类 C_i 在 E 中出现的概率。

定义7.32 在决策树的生成过程中, 每一次面向属性的测试都必须使属性选择函数ASF的值达到最大。

定义7.33 ASF函数定义如下:

$$ASF(A_i) = \frac{f(A_i) \bullet g(A_i)}{h(A_i)}$$

其中, $f(A_i)$ 表示属性 i 的效益; $g(A_i)$ 表示属性 i 的偏置值; $h(A_i)$ 表示属性 i 的代价; $1 \leq i \leq k$ 。

(1) 右式给出了 $f(A_i)$ 的定义。其中 U^I 表示有用信息; N^I 表示无用信息; T^I 表示总信息 $= U^I + N^I$ 。

$$f() = \frac{U^I}{T^I} = 1 - \frac{N^I}{T^I}$$

令：

$$\begin{aligned}\Delta I &= H(T^I) - H(N^I) \\ -\Delta I &= H(N^I) - H(T^I) = \log_2(N^I) - \log_2(T^I) = \log_2 \left[\frac{N^I}{T^I} \right] \\ 2^{-\Delta I} &= \left[\frac{N^I}{T^I} \right] = 1 - \left[\frac{U^I}{T^I} \right] = 1 - f() \\ \Delta I &= \text{Gain}(A_i, E) = I(E) - \text{Ent}(A_i, E) \\ f() &= 1 - 2^{-\Delta I} \\ I(E) &= -\sum_{j=1}^k P_j \log_2 P_j, P_j = \frac{|E \cap C_j|}{|E|} \bullet CT_j \\ \text{Ent}(A_i, E) &= \sum_{i=1}^v \frac{|E_i|}{|E|} I(E_i) \\ f() &= 1 - 2^{-\text{Gain}()}\end{aligned}$$

为：

$$(3) h(A_i) \text{ 定义为: } g(A_i) = C^b_i$$

$$h(A_i) = C^c_i + 1$$

从而得到的 ASF 表达式：

$$ASF() = \frac{(1 - 2^{-\text{Gain}()}) \bullet C^b}{C^c + 1}$$

我们的决策树生成算法 GSD 是 Quinlan 的 C4.5 算法的一个修改版。修改体现在输入及属性选择等两个方面。GSD 的输入采用预处理算法 PPD 输出的指定概念层次的训练例子集；并用属性选择函数 ASF 代替了 C4.5 的属性选择标准。

算法 7.11 决策树生成算法 GSD。

输入：训练集

输出：决策树，规则集

- (1) 在 Gain、Gain Ratio 及 ASF 中选择出标准 CR；
- (2) 本如果所有数据项属于同一类，则决策树是一个标有该类标志的叶子；
- (3) 否则，用标准 CR 选取最佳测试属性把数据项分为子集；
- (4) 递归地调用 (2)、(3) 过程为每一个子集生成一棵决策树；
- (5) 生成决策树并转化为规则集。

7.7.9 实验结果

BSDT 算法用到了下列数据集(见表 7.3)。

表 7.3 实验数据集

数据集	#训练例子数	#属性数	#类别数	#测试集例子数
anneal	898	38	5	-

breast-cancer	699	10	2	–
credit	490	15	2	200
genetics	3,190	60	3	–
glass	214	9	6	–
heart	1,395	16	2	–
hypo	2,514	29	5	1,258
letter	15,000	16	26	5,000
sonar	208	60	2	–
soybean	683	35	19	–
voting	300	16	2	135
diabetes	768	8	2	–

这些数据集中的一部分来源于 UCI 测试库 (<ftp://ics.uci.edu/pub/machine-learning-database>)。在测试数据集的时候，我们采取了下述措施：

将没有给定测试集的数据集 DS（如 anneal、breast-cancer 等）利用随机选取的办法分为测试集 TS 与学习集 LS 并令 $TS = DS \times 10\%$ ；

- 用 C4.5 算法在 LS 上用 10 次交叉验证生成分类器 C0，在 TS 上测试后得到的预测准确度 P_0 ；

表 7.4 预测准确度比较

数据集	P_0	P_1	P_2
Anneal	96.9 ± 10.4	97.1 ± 9.7	97.8 ± 10.8
breast-cancer	95.7 ± 2.1	96.2 ± 1.7	97.2 ± 2.5
credit	84.8 ± 2.5	87.4 ± 1.8	89.5 ± 2.1
genetics	98.7 ± 4.4	98.7 ± 4.4	98.7 ± 4.4
glass	68.9 ± 9.2	75.2 ± 6.5	75.2 ± 6.5
heart	77.8 ± 4.3	79.6 ± 3.9	79.8 ± 3.7
hypo	93.2 ± 4.2	92.7 ± 3.9	92.7 ± 3.9
letter	88.4 ± 9.8	93.7 ± 8.7	93.7 ± 8.7
sonar	65.4 ± 7.1	74.7 ± 12.1	74.7 ± 12.1
soybean	78.9 ± 5.9	84.8 ± 6.5	84.8 ± 6.5
voting	95.8 ± 1.3	95.9 ± 1.6	95.9 ± 1.6
diabetes	74.3 ± 3.0	76.1 ± 3.1	77.2 ± 2.3

- 用 BSDT 算法的 gain ratio 属性选取标准在 LS 上用 10 次交叉验证生成分类器 C1，在 TS 上测试后得到的预测准确度 P_1 ；
- 用 BSDT 算法的 ASF 属性选取标准在 LS 上用 10 次交叉验证生成分类器 C2，在 TS 上测试后得到的预测准确度 P_2 ；
- 每种算法都进行了 15 次试验的平均值，以期得到稳定的预测模型；
- 将 P_0 、 P_1 及 P_2 的结果进行比较，结果见表 7.4。

从表 7.4 可以看出：所有的数据集经 BSDT 算法生成的决策树分类预测模型进行分类的预测准确度都改善了。由代价系数及偏置系数定义的 ASF 函数，预测准确度得到明显提高。实验结果表明：ASF 函数与 BSDT 算法是行之有效的。

我们在 BSDT 算法中实现了一种可以集成多种决策树学习算法的方法。它是一种集成多种机器学习算法的有效途径，并保证最佳的表示偏置与过程偏置的选择，从而支持多策略学习算法。

7.8 归纳学习的计算理论

学习的计算理论主要研究学习算法的样本复杂性和计算复杂性。本节重点讨论 Gold 学习理论和 Valiant 学习理论，并将它们进行比较。

对于建立机器学习科学，学习的计算理论非常重要，否则无法识别学习算法的应用范围，也无法分析不同方法的可学习性。收敛性、可行性和近似性是本质问题，它们要求学习的计算理论给出一种令人满意的学习框架，包括合理的约束。这方面的早期成果主要是基于 Gold 框架。在形式语言学习的上下文中，Gold 引入收敛的概念，有效地处理了从实例学习的问题。学习算法允许提出许多假设，无须知道什么时候它是正确的，只要确认某点它的计算是正确的假设。由于 Gold 算法的复杂性很高，因此这种风范并没有在实际学习中得到应用。

基于 Gold 学习框架，Shapiro 提出了模型推理算法研究形式语言与其解释之间的关系，也就是形式语言的语法与语义之间的关系。模型论把形式语言中的公式、句子理论和它们的解释——模型，当作数学对象进行研究。Shapiro 模型推理算法只要输入有限的事实就可以得到一种理论输出[Shapiro 1981]。

1984 年 Valiant 提出一种新的学习框架[Valiant 1984]。它仅要求与目标概念具有高概率的近似，而并不要求目标概念精确的辨识。Kearns、Li、Pitt 和 Valiant 对可以表示为布尔公式的概念给出了一些新的结果。Haussler 应用 Valiant 框架分析了变型空间和归纳偏置问题，并给出了样本复杂性的计算公式。

7.8.1 Gold 学习理论

Gold 的语言学习理论研究引入两个基本概念，即极限辨识和枚举辨识，这对早期的归纳推理的理论研究起了非常重要的作用[Gold 1967]。

极限辨识把归纳推理看作一种无限过程，归纳推理方法的最终或极限行为可以看作是它的成功标准。假设 M 是一种归纳推理方法，它企图正确地描述未知规则 R 。假设 M 重复运行， R 的实例集合则愈来愈大，形成 M 推测的无限序列 g_1, g_2, \dots 。如果存在某个数 m ，使得 g_m 是 R 的正确描述，

$$g_m \models g_{m+1} \models g_{m+2} \models \dots,$$

那么 M 在这个实例序列的极限正确地辨识 R 。 M 可以看作对未知规则 R 学习愈来愈多，成功地修改它关于 R 的推测。如果有限次后 M 停止修改它的推测，最后的推测就是 R 的正确描述，那么在这个实例序列的极限 M 正确地辨识 R 。注意， M 不能确定它是否会收敛到一个正确的假设，因为新的数据与当前的推测是否会发生矛盾并不知道。

枚举辨识是第一种方法推测多项式序列的抽象，即对可能的规则空间进行系统搜索，直到发现与迄今为止的所有数据相一致的推测。假设规定了规则的具体领域，有一个描述枚举，即 d_1, d_2, d_3, \dots ，以致于领域中的每一条规则在枚举中有一种或多种描述。给定一条规则的

某个实例集合，枚举辨识方法将通过这个表，找到第一个描述 d_i ，即与给定的实例相容，那么推测为 d_i 。这种方法不能确定是否会达到正确的极限辨识。如果实例表示和相容关系满足下面两个条件，那么枚举方法保证极限辨识该领域中的全部规则：

- (1) 一个正确假设总是与给定的实例相容。
- (2) 任何不正确的假设与实例足够大的集合或与全部集合不相容。

为了枚举方法是可计算的，枚举 d_1, d_2, d_3, \dots 必须是可计算的，它必须能够计算给定的描述与给定的实例集合是相容的。

算法 7.12 枚举辨识算法

输入：

- 一组表达式的集合 $E = e_1, e_2, \dots$ 。
- 谕示(oracle) TE 提供足够的目标实例集。
- 排序信息的谕示 LE。

输出：

一系列假设断言 H_1, H_2, \dots ，每个假设 H_i 都在 E 中，并与第 i 个实例一致。

过程：

- (1) 初始化, $i \leftarrow 1$;
- (2) $examples \leftarrow \text{emptyset}$;
- (3) Loop:
 - 3.1 调用 TE(), 将 example 加到集合 examples;
 - 3.2 While LE($e_i, +x$) = no, $+x$, 或者
LE($e_i, -x$) = yes, 对反例集 $-x$,
 $i \leftarrow i + 1$;
- (4) 输出 e_i 。

7.8.2 模型推理系统

模型推理问题是科学家所面临的问题抽象，他们在具有固定概念框架的某种领域里工作，进行试验，试图找到一种理论可以解释他们的结果。在这种抽象中研究的领域是对给定的一阶语言 L 某种未知模型 M 的领域，实验是检测 M 中 L 语句的真值，目标是寻找一组正确假设，它们包含全部正确的可测试的句子。

L 语句分成两个子集：观测语言 L_o 和假设语言 L_h 。假设

$$\square \in L_o \subset L_h \subset L'$$

其中 \square 是空语句。那么模型推理问题可以定义如下：假设给定一阶语言 L 和两个子集：观测语言 L_o 和假设语言 L_h 。另外对 L 的未知模型 M 给定一种处理机制 oracle。模型推理问题是寻找 M 的一种有限的 L_o —— 完备公理化。

求解模型推理问题的算法称为模型推理算法。模型 M 的枚举是一个无限序列 F_1, F_2, F_3, \dots ，其中 F_i 是关于 M 的事实， L_o 的每个语句 α 发生在事实 $F_i = \langle \alpha, V \rangle, i \geq 0$ 。模型推理算法一次读入给定观测语言 L_o 的模型的一种枚举，一个事实，产生假设语言 L_h 的语句的有限集称为算法的推测。一种枚举模型推理算法如下。

算法 7.13 枚举模型推理算法

h 是整个递归函数。

设 S_{false} 为 $\{\square\}$, S_{true} 为 $\{\}$, k 为 0。

repeat

 读入下一个事实 $F_n = \langle \alpha, V \rangle$,

α 加到 S_v ,

 while 有一个 $\alpha \in S_{false}$ 以致于 $T_k \vdash_n \alpha$

 或有一个 $\alpha_i \in S_{true}$ 以致于 $T_k \not\vdash_{n(i)} \alpha_i$ do

$k = k+1$,

 输出 T_k 。

forever

上面算法中 $T \vdash_n \alpha$ 表示在推导 n 步或少于 n 步时, 假设语句 T 可以推导出 α 。 $T \not\vdash_{n(i)} \alpha$

表示在推导 n 步或少于 n 步时, 假设语句 T 不能推出 α 。推导中假设是单调的。Shapiro 证明这种算法是极限辨识。这种算法功能强且灵活, 可以从事实推出理论, 是一种递增算法。

7.8.3 Valiant 学习理论

Valiant 认为一个学习机必须具备下列性质:

- (1) 机器能够证明地学习所有类的概念。更进一步, 这些类可以特征化。
- (2) 对于通用知识概念类是合适的和不平常的。
- (3) 机器演绎所希望的程序的计算过程要求在可行的步数内。

学习机由学习协议和演绎过程组成。学习协议规定从外部获得信息的方法。演绎过程是一种机制, 学习概念的正确识别算法是演绎的。从广义来看, 研究学习的方法是规定一种可能的学习协议, 使用这种协议研究概念类, 识别程序可以在多项式时间内演绎。具体协议允许提供两类信息。第一种是学习者对典型数据的访问, 这些典型数据是概念的正例。要确切地说, 假设这些正例本质上有一种任意确定的概率分布。调用子程序 EXAMPLES 产生一种这样的正例。产生不同例子的相对概率是分布确定的。第二个可用的信息源是 ORACLE。在最基本的版本中, 当提交数据时, 它将告诉学习该数据是否是概念的正例示。

假设 X 是实例空间, 一个概念是 X 的一个子集。如果实例在概念中则为正例, 否则为反例。概念表示是一种概念的描述, 概念类是一组概念表示。学习模型是概念类的有效的可学习性。Valiant 学习理论仅要求对目标概念的很好近似具有极高的概率。允许学习者产生的概念描述与目标概念有一个小的偏差 ϵ , 它是学习算法的一个输入参数。并且, 允许学习者失败的概率为 δ , 这也是一个输入参数。两种概念之间的差别采用在实例空间 X 的分布概率 D 来评测:

$$\text{diff}_D(c_1, c_2) = \sum_{x \in X, c_1(x) \neq c_2(x)} D(x) \quad (7.37)$$

根据协议, 一个概念类 C 是可学习的当且仅当有一种算法 A , 使用协议, 对所有的目标概念表示 $c^* \in C$ 和全部分布 D ,

- (1) 执行时间是与 $\frac{1}{\epsilon}, \frac{1}{\delta}, c^*$ 数目和其它相关参数有关的多项式。

(2) 输出 C 中的概念 c 具有概率 $1-\delta$,

$$\text{diff}_D(c, c^*) < \epsilon$$

Valiant 学习理论中, 有两种学习复杂性测度。一种是样本复杂性。这是随机实例的数目, 用以产生具有高的概率和小的误差。第二种性能测度是计算复杂性, 定义为最坏情况下以给定数目的样本产生假设所要求的计算时间。

设 L 是学习算法, C 是例示空间 X 上的一类目标概念。对于任意的 $0 < \epsilon, \delta < 1$, $S_c^L(\epsilon, \delta)$ 表示最小的样本数 m 使任意目标概念 $c \in C$, X 上任意分布, 给定 m 个 c 的随机样本, L 产生一个假设, 其概率至少为 $1-\delta$, 误差最大为 ϵ 。 $S_c^L(\epsilon, \delta)$ 称为目标类 c 的 L 样本复杂性。

下面我们讨论两种学习算法的样本复杂性。

1. 学习合取概念的分类算法

该算法的内容如下:

- (1) 对于给定的样本找出每个属性最小的主原子。这些原子合取形成假设 h 。
- (2) 如果 h 中不包含反例, 那么返回 h , 否则得到结果, 认为样本与任何纯合取概念不一致。

Haussler 分析这种算法, 得出其样本复杂性 $S_c^L(\epsilon, \delta)$ 为

$$C_0(\log(\frac{1}{\delta}) + n)/\epsilon \leq S_c^L(\epsilon, \delta) \leq C_1(\log(\frac{1}{\delta}) + n \log(\frac{1}{\epsilon}))/\epsilon \quad (7.38)$$

其中 C_0, C_1 是正的常数。

2. 学习纯合取概念的贪婪算法

算法 7.14 学习纯合取概念的贪婪算法

- (1) 对于给定的样本找出每个属性最小的主原子。
- (2) 开始纯合取假设 h 为空, 当样本中有反例时则做
 - 2.1 在所有属性中, 找到最小的主原子, 它删除最多的反例, 把它加到 h , 如果没有最小主原子可以删除任何反例, 则循环结束。
 - 2.2 从样本中去掉已删除的反例。
- (3) 如果没有反例则返回 h , 否则报告样本与任何纯合取概念不一致。

Haussler 在文章 [Haussler 1988a] 中详细讨论了该算法的样本复杂性问题, 并给出如下结果:

$$C_0(\log(\frac{1}{\delta}) + s \log(\frac{n}{s}))/\epsilon \leq S_c^L(\epsilon, \delta) \leq C_1(\log(\frac{1}{\delta}) + s(\log(\frac{sn}{\epsilon}))^2)/\epsilon \quad (7.39)$$

其中 C_0, C_1 是正常数, s 为纯合取概念中最大的原子数。

由上看出, Valiant 学习理论, 仅要求学习算法产生的假设能以高的概率很好接近目标概念, 并不要求精确地辨识目标概念。这种学习理论是“几乎大体正确”辨识, 有时简称为 PAC(Probably Approximately Correct)理论。Valiant 学习理论比 Gold 学习理论更有实际意义。

习 题

1. 何谓归纳学习, 其主要特点是什么?
2. 通过实例, 说明选择型和构造型泛化规则的用途。
3. 什么是决策树学习中的偏置问题, 简述几种偏置学习算法。

4. 何谓假设空间？假设空间中的各假设间存在什么关系？
5. AQ 学习方法遵从的一般归纳推理模式是什么样的？为什么说 AQ 学习的过程就是搜索假设空间的过程？
6. 结合规则空间排序示意图，描述变型空间方法的基本思想。
6. 给出候选项删除算法用于下列数据集的过程：
 - 正实例： a) object(red, round, apple)
 - b) object(green, round, mango)
 - 负实例： a) object(red, large, banana)
 - b) object(green, round, guava)
7. 阐述决策树学习方法及其适用场合。
8. 在构造决策树的过程中，测试属性的选取采用什么原则？如何实现？
9. 叙述 ID3 算法的基本思想和建树步骤。
10. 给定下面的数据，回答问题。

StudiedHard	HoursSelptBefore	Breakfast	GotA
No	5	Eggs	No
No	9	Eggs	No
Yes	6	Eggs	No
No	6	Bagel	No
Yes	9	Bagel	Yes
Yes	8	Eggs	Yes
Yes	8	Cereal	Yes
Yes	6	Cereal	Yes

- (1) GotA 的初始熵为多少？
- (2) 决策树算法（ID3）会选择哪个属性作为根结点？
- (3) 构造该决策树。
11. C4.5 算法对 ID3 算法的改进体现在哪些方面。
12. 何谓学习算法的样本复杂性和计算复杂性？
13. 为什么 Valiant 学习理论比 Gold 学习理论更有实际意义？

第八章 支持向量机

Vapnik 等人从 20 世纪 60 年代开始就研究有限样本的机器学习问题。经过几十年的研究，到 90 年代中期终于形成了一个较完整的理论体系，即统计学习理论[Vapnik 1995]。近几年来，在统计学习理论的基础上又发展出一种新的机器学习方法，即支持向量机 (Support Vector Machine, SVM)。支持向量机是建立在计算学习理论的结构风险最小化原则之上，其主要思想是针对两类分类问题，在高维空间中寻找一个超平面作为两类的分割，以保证最小的分类错误率[Vapnik 1997]。SVM 一个重要的优点是可以处理线性不可分的情况。

8.1 统计学习问题

8.1.1 经验风险

学习的目的是根据给定的训练样本求系统输入输出之间的依赖关系。学习问题可以一般地表示为变量 y 与 x 之间存在的未知依赖关系，即遵循某一未知的联合概率 $F(x, y)$ 。机器学习问题就是根据 n 个独立同分布观测样本：

$$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n) \quad (8.1)$$

在一组函数 $\{f(x, w)\}$ 中，求一个最优的函数 $f(x, w_0)$ ，对依赖关系进行估计，使期望风险

$$R(w) = \int L(y, f(x, w)) dF(x, y), \quad (8.2)$$

最小。其中， $\{f(x, w)\}$ 称作预测函数集， w 为函数的广义参数。 $\{f(x, w)\}$ 可以表示任何函数集。 $L(y, f(x, w))$ 为由于用 $f(x, w)$ 对 y 进行预测而造成的损失。不同类型的学习问题有不同形式的损失函数。

在传统的学习方法中，采用了所谓的经验风险最小化 (Empirical Risk Minimization, ERM) 准则，即用样本定义经验风险

$$R_{emp}(w) = \frac{1}{l} \sum_{i=1}^l L(y_i, f(x_i, w)) \quad (8.3)$$

机器学习就是要设计学习算法使 $R_{emp}(w)$ 最小化，作为对 (8.2) 式的估计。

8.1.2 VC 维

统计学习理论是关于小样本进行归纳学习的理论。其中一个重要的概念是 VC 维 (Vapnik-Chervonenkis Dimension)。模式识别方法中 VC 维的直观定义是：对于一个指示函数集，如果存在 h 个样本能够被函数集里的函数按照所有可能的 2^h 种形式分开，则称函数集能够把 h 个样本打散。函数集的 VC 维就是它能打散的最大样本数目 h 。若对任意数目的样本都有函数能将它们打散，则函数集的 VC 维是无穷大。有界实函数的 VC 维可以通过用一定的阈值将它转化成指示函数来定义。

VC 维反映了函数集的学习能力。一般而言，VC 维越大则学习机器越复杂，学习容量就越大。目前尚没有通用的关于任意函数集 VC 维计算的理论，只对一些特殊的函数集知道其函

数维。例如在 n 维实数空间中线性分类器和线性实函数的 VC 维是 $n+1$ ，而 $f(x, \alpha) = \sin(\alpha x)$ 的 VC 维则为无穷大。如何用理论或实验的方法计算其 VC 维是当前统计学习理论中有待研究的一个问题。

8.2 学习过程的一致性

8.2.1 学习一致性的经典定义

为了从有限的观察中构造学习算法，我们需要一种渐近理论，刻画学习过程一致性的必要和充分条件。

定义 8.1 经验风险最小一致性原理

对于指示函数集 $L(y, w)$ 和概率分布函数 $F(y)$ ，如果下面两个序列概率地收敛到同一极限（见图 8.1），则称为经验风险最小一致性：

$$R(w_l) \xrightarrow[l \rightarrow \infty]{P} \inf_{w \in \Lambda} R(w) \quad (8.4)$$

$$R_{emp}(w_l) \xrightarrow[l \rightarrow \infty]{P} \inf_{w \in \Lambda} R(w) \quad (8.5)$$

换句话说，如果经验风险最小化方法是一致性，那么它必须提供一个函数序列 $L(y, w_l)$ ， $l=1, 2, \dots$ ，使得期望风险和经验风险收敛到一个可能最小的风险值。（8.4）式判定风险收敛到最好的可能值。（8.5）式人们可以根据经验风险判定估计风险可能的最小值。

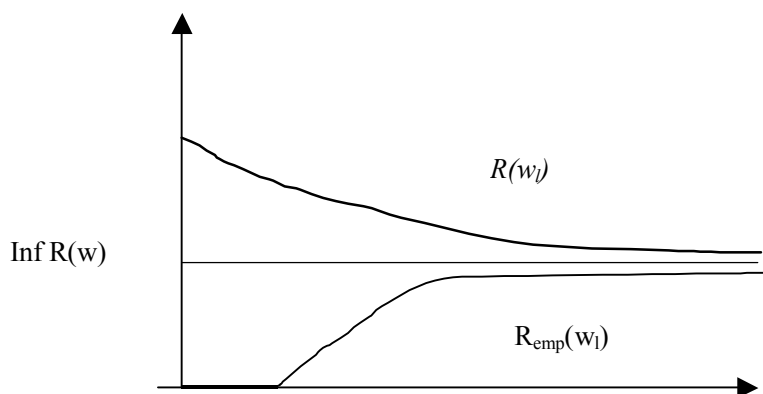


图 8.1 学习过程一致性

8.2.2 学习理论的重要定理

Vapnik 和 Chervonenkis 于 1989 年提出了学习理论的重要定理如下：

定理 8.1 设 $L(y, w)$, $w \in \Lambda$ 是满足下列条件的函数集

$$A \leq \int L(y, w) dF(y) \leq B \quad (A \leq R(w) \leq B). \quad (8.6)$$

那么对于经验风险最小是一致的，在下面情况下：

$$\lim_{l \rightarrow \infty} P\{\sup_{w \in \Lambda} R(w) - R_{emp}(w) > \varepsilon\} = 0, \quad \forall \varepsilon > 0. \quad (8.7)$$

经验风险 $R_{emp}(w)$ 在整个函数集 $L(y, w), w \in \Lambda$ 上一致地收敛到实际风险 $R(w)$ 。我们称这类一致性收敛为单侧一致性收敛。

8.2.3 VC 熵

定义 8.2 设 $A \leq L(y, w) \leq B, w \in \Lambda$ 是界限损耗函数集。使用这个函数集和训练集 z_1, \dots, z_l ，可以构造下列 l 维向量：

$$q(w) = (L(z_1, w), \dots, L(z_l, w)), \quad w \in \Lambda. \quad (8.8)$$

这个向量集属于 l 维立方体，并且在品质 C 上具有有限最小 ε 网格。设 $N = N^\Lambda(\varepsilon; z_1, \dots, z_l)$ 是向量集 $q(w), w \in \Lambda$ 最小 ε 网格元素的数目。

注意， $N^\Lambda(\varepsilon; z_1, \dots, z_l)$ 是随机变量，因为它是使用随机向量 z_1, \dots, z_l 构造的。随机值 $N^\Lambda(\varepsilon; z_1, \dots, z_l)$ 的对数

$$H^\Lambda(\varepsilon; z_1, \dots, z_l) = \ln N^\Lambda(\varepsilon; z_1, \dots, z_l)$$

被称作函数集 $A \leq L(y, w) \leq B, w \in \Lambda$ 对于训练样本 z_1, \dots, z_l 随机 VC 熵。随机 VC 熵的期望值：

$$H^\Lambda(\varepsilon; l) = EH^\Lambda(\varepsilon; z_1, \dots, z_l)$$

被称作函数集 $A \leq L(y, w) \leq B, w \in \Lambda$ 对于训练样本数 l 的 VC 熵。这里，期望值取乘积测度 $F(z_1, \dots, z_l)$ 。

定理 8.2 双侧一致收敛的必要和充分条件是下式成立：

$$\lim_{l \rightarrow \infty} \frac{H^\Lambda(\varepsilon, l)}{l} = 0, \quad \forall \varepsilon > 0 \quad (8.9)$$

换句话说，VC 熵与观察数的比值应随观察数增加而减小到 0。

推论 8.1 在指示函数集 $L(y, w), w \in \Lambda$ 一定的可测条件下，双侧一致收敛的必要和充分条件是：

$$\lim_{l \rightarrow \infty} \frac{H^\Lambda(l)}{l} = 0,$$

它是 (8.9) 式的特例。

定理 8.3 对于整个界限函数集 $L(y, w), w \in \Lambda$ ，为了使经验方法单侧一致收敛到它们的期望值，其必要和充分的条件是对于任何正的 δ, η 和 ε ，存在函数集 $L^*(y, w^*), w^* \in \Lambda^*$ 满足

$$\begin{aligned} L(y, w) - L^*(y, w^*) &\geq 0, \forall y, \\ \int (L(y, w) - L^*(y, w^*)) dF(y) &\leq \delta. \end{aligned} \quad (8.10)$$

对于样本数 l ， $L^*(y, w^*), w^* \in \Lambda^*$ 的 ε 熵下式成立：

$$\lim_{l \rightarrow \infty} \frac{H^{\Lambda^*}(\varepsilon, l)}{l} < \eta. \quad (8.10)$$

在进行学习理论研究中，可以分为三个里程碑，即依次根据以下三种方法来定义经验风险最小化原理的一致性的充分条件：

(1) 运用 VC 熵来定义：

$$\lim_{l \rightarrow \infty} \frac{H^\Lambda(l)}{l} = 0 \quad (8.11)$$

(2) 运用退火熵来定义：

$$\lim_{l \rightarrow \infty} \frac{H^{\Lambda_{ann}}(l)}{l} = 0 \quad (8.12)$$

其中退火 VC 熵为

$$H^{\Lambda_{ann}}(l) = \ln EN^\Lambda(z_1, \dots, z_l).$$

(3) 运用增长函数来定义：

$$\lim_{l \rightarrow \infty} \frac{G^\Lambda(l)}{l} = 0 \quad (8.13)$$

其中增长函数为

$$G^\Lambda(l) = \ln \sup_{z_1, \dots, z_l} N^\Lambda(z_1, \dots, z_l).$$

8.3 结构风险最小归纳原理

统计学习理论系统地研究了对于各种类型的函数集、经验风险和实际风险之间的关系，即泛化的界限[Vapnik 1995]。关于两类分类问题有如下结论：对指示函数集中的所有函数（包括使经验风险最小的函数），经验风险 $R_{emp}(w)$ 和实际风险 $R(w)$ 之间以至少 $1-\eta$ 的概率满足如下的关系[Burges 98]：

$$R(w) \leq R_{emp}(w) + \sqrt{\frac{h(\ln(2l/h) + 1) - \ln(\eta/4)}{l}} \quad (8.14)$$

其中 h 是函数集的 VC 维, l 是样本数, η 是满足 $0 \leq \eta \leq 1$ 的参数。

由此可见, 统计学习的实际风险 $R(w)$ 由两部分组成: 一是经验风险 (训练误差) $R_{emp}(w)$, 另一部分称为置信界限 (VC confidence)。置信界限反映了真实风险和经验风险差值的上确界, 反映了结构复杂所带来的风险, 它和学习机器的 VC 维 h 及训练样本数 l 有关。(8.14) 式可以简单地表示为:

$$R(w) \leq R_{emp}(w) + \Phi(h/l) \quad (8.15)$$

由(8.14)式右端第二项可知, $\Phi(h/l)$ 随 h 加大而增长。因此, 在有限训练样本下, 学习机器的复杂性越高, VC 维越高, 则置信界限越大, 就会导致真实风险与经验风险之间的可能的差别越大。

请注意, 这里的泛化界限是对于最坏情况的结论。在很多情况下是较松的, 尤其当 VC 维较高时更是如此。文献[Burges 1998] 指出, 当 $h/l > 0.37$ 时, 这个界限肯定是松弛的。当 VC 维无穷大时, 这个界限就不再成立。

为了构造适合于对小样本学习的归纳学习原理, 可以通过控制学习机器的泛化能力来达到此目的。对于完全有界非负函数, $0 \leq L(z, w) \leq B$, $w \in \Lambda$ (Λ 是抽象参数集合), 以至少 $1 - \eta$ 的概率满足以下不等式,

$$R(w_l) \leq R_{emp}(w) + \frac{B\varepsilon}{2} \left(1 + \sqrt{1 + \frac{4R_{emp}(w_l)}{B\varepsilon}}\right), \quad (8.15)$$

其中, 对于无界限函数集合, 学习机器的泛化能力的界限是

$$R(w_l) \leq \frac{R_{emp}(w_l)}{(1 - a(p)\tau\sqrt{\varepsilon})_+} \quad (8.16)$$

$$a(p) = \sqrt[p]{\frac{1}{2} \left(\frac{p-1}{p-2}\right)^{p-1}}$$

其中,

$$\varepsilon = 2 \frac{\ln N - \ln \eta}{l}$$

运用以上的公式可以控制基于固定数目的经验样本之上的风险函数的最小化的过程。控制最小化过程的有几种方法。一是最小化经验风险。根据上面的公式可以看出, 当风险的上界随着经验风险的值的减少而减少。二是最小化 (8.15) 式中右边的第二项, 将样本的数目与 VC 维作为控制变量。后者适合于样本数较小的情况。

设函数集 $L(z, w)$ 具有嵌套函数子集 $S_k = \{L(z, w), w \in \Lambda_k\}$ 的结构组成, 如图 8.2 所示

$$S_1 \subset S_2 \subset \cdots \subset S_n, \quad (8.17)$$

其中结构元素满足以下性质:

(1) 每个函数集 S_k 的拥有一个有限的 VC 维 h_{kk} ;

$$h_1 \leq h_2 \leq \dots, \leq h_n, \dots$$

(2) 结构中的任何元素 S_k 包含性质:

整个界限函数集满足

$$0 \leq L(z, w) \leq Bk; \quad \alpha \in \Lambda_k$$

或者对于某种 (p, τ_k) , 函数集满足下列不等式:

$$\sup_{w \in \Lambda_k} \frac{(\int L^p(z, w) dF(z))^{\frac{1}{p}}}{\int L(z, w) dF(z)} \leq \tau_k \quad p > 2 \quad (8.18)$$

我们称这种结构为可同伦结构。

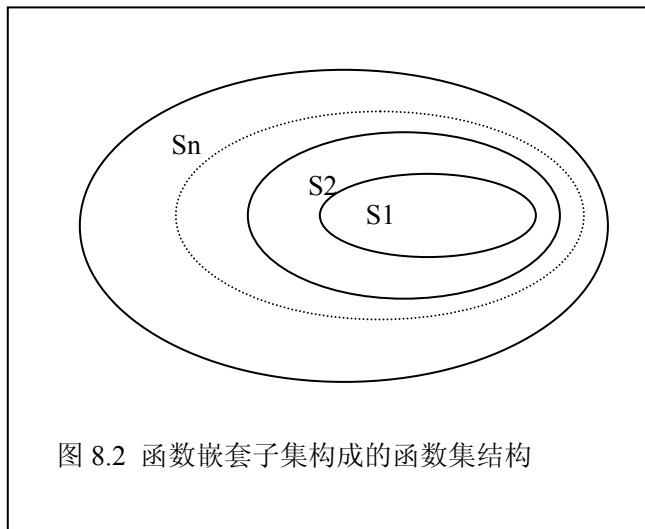


图 8.2 函数嵌套子集构成的函数集结构

为了选择合适的 S_k 作为学习函数, 可以将 (8.14) 式右边划分为两个部分, 左边项为经验风险; 右边项为置信区。如果给定样本数目 l , 那么, 随着 VC 维数目 h 的增加, 经验风险逐渐变小, 而置信区逐渐递增。

如图 8.3 所示, 综合考虑经验风险与置信区的变化, 可以求得最小的风险边界, 它所对应的函数集的中间子集 S^* 可以作为具有最佳泛化能力的函数集合。

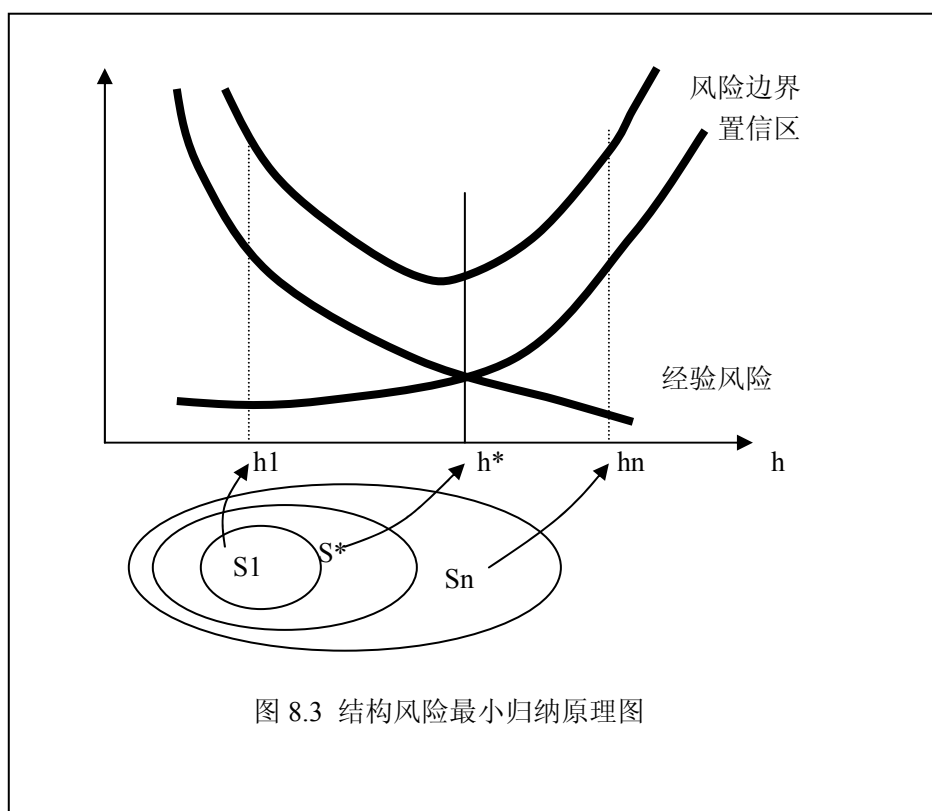


图 8.3 结构风险最小归纳原理图

8.4 支持向量机

支持向量机（Support Vector Machine）是近几年发展起来的新型的通用知识发现方法，在分类方面具有良好的性能。

8.4.1 线性可分

假设存在训练样本 $(x_1, y_1), \dots, (x_l, y_l)$, $x \in R^l$, $y \in \{+1, -1\}$, l 为样本数, n 为输入维数, 在线性可分的情况下就会有一个超平面使得这两类样本完全分开。该超平面描述为:

$$(w \cdot x) + b = 0 \quad (8.19)$$

其中, “ \cdot ” 是向量点积。分类如下:

$$w \cdot x_i + b \geq 0, \quad \text{对于 } y_i = +1$$

$$w \cdot x_i + b < 0, \quad \text{对于 } y_i = -1$$

w 是超平面的法线方向, $\frac{w}{\|w\|}$ 为单位法向量, 其中 $\|w\|$ 是欧氏模函数。

如果训练数据可以无误差地被划分, 以及每一类数据与超平面距离最近的向量与超平面之间的距离最大则称这个超平面为最优超平面 (如图 8.4 所示)。

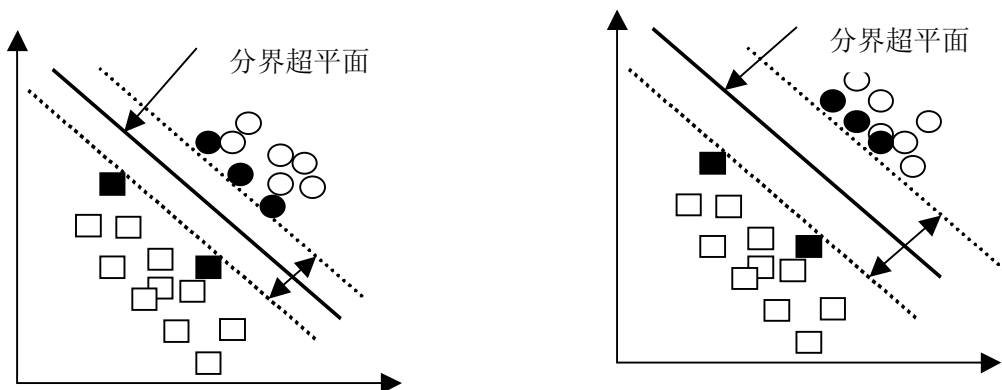


图 8.4 最优超平面

在线性可分情况下, 求解最优超平面, 可以看成解二次型规划的问题。对于给定的训练样本, 找到权值 w 和偏移 b 的最优值, 使得权值代价函数最小化:

$$\min \Phi(w) = \frac{1}{2} \|w\|^2 \quad (8.20)$$

满足约束条件:

$$y_i(w \bullet x_i + b) - 1 \geq 0, \quad i = 1, 2, \dots, l \quad (8.21)$$

优化函数 $\Phi(w)$ 为二次型, 约束条件是线性的, 因此是个典型的二次规划问题, 可由 Lagrange 乘子法求解。引入 Lagrange 乘子 $\alpha_i \geq 0, i = 1, 2, \dots, l$:

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^l \alpha_i \{y_i(x_i \bullet w + b) - 1\} \quad (8.22)$$

这里, L 的极值点为鞍点, 可取 L 对 w 和 b 的最小值 $w=w^*, b=b^*$, 以及对 α 的最大值 $\alpha=\alpha^*$ 。对 L 求导可得:

$$\frac{\partial L}{\partial b} = \sum_{i=1}^l y_i \alpha_i = 0 \quad (8.23)$$

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^l y_i \alpha_i x_i = 0 \quad (8.24)$$

其中, $\frac{\partial L}{\partial w} = \left(\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \dots, \frac{\partial L}{\partial w_l} \right)$ 。

于是, SVM 通过求解二次规划, 得到对应的 α^* 和 w^* :

$$w^* = \sum_{i=1}^l \alpha_i^* y_i x_i \quad (8.25)$$

以及最优的超平面(见图 8.4)。

经过变换线性可分条件下的原问题成为对偶问题, 求解如下的极大化:

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j y_i y_j x_i \bullet x_j = \Gamma \bullet I - \frac{1}{2} \Gamma \bullet D \Gamma \quad (8.26)$$

满足约束:

$$\sum_{i=1}^l y_i \alpha_i = 0, \quad \alpha_i \geq 0, i = 1, 2, \dots, l \quad (8.27)$$

其中, $\Gamma = (\alpha_1, \alpha_2, \dots, \alpha_l)$, $I = (1, 1, \dots, 1)$, D 是 $l \times l$ 的对称矩阵, 各个单元为:

$$D_{ij} = y_i y_j x_i \bullet x_j \quad (8.28)$$

在对这类约束优化问题的求解和分析中, Karush-Kuhn-Tucker(KKT)条件将起重要作用。如(8.27)式这样的问题, 其解必须满足:

$$\alpha_i \{y_i (w \bullet x_i + b) - 1\} = 0, \quad i = 1, 2, \dots, l \quad (8.29)$$

从(8.25)式看到, 那些 $\alpha_i = 0$ 的样本对于分类问题不起什么作用, 只有 $\alpha_i > 0$ 的样本对 w^* 起作用, 从而决定分类结果。这样的样本定义为支持向量。

所求向量中, α^* 和 w^* 可以被训练算法显式求得。选用一个支持向量样本 x_i , 可以这样求得 b^* :

$$b^* = y_i - w \bullet x_i \quad (8.30)$$

对输入样本 x 测试时, 计算下式

$$d(x) = x \bullet w^* + b^* = \sum_{i=1}^l y_i \alpha_i^* (x \bullet x_i) + b^* \quad (8.31)$$

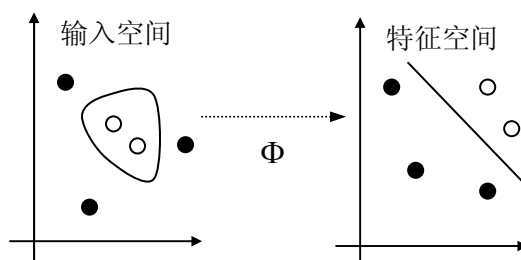
根据 $d(x)$ 的符号来确定 x 的归属。

8.4.2 线性不可分

线性可分的判别函数建立在欧氏距离的基础上, 即 $K(x_i, x_j) = x_i \bullet x_j = x_i^T x_j$ 。对非线性问题, 可以把样本 x 映射到某个高维特征空间 H (见图 8.5), 并在 H 中使用线性分类器, 换句话说, 将 x 做变换 $\Phi: R^d \rightarrow H$:

$$x \rightarrow \Phi(x) = (\phi_1(x), \phi_2(x), \dots, \phi_{l(x)}(x), \dots)^T \quad (8.32)$$

其中, $\phi_l(x)$ 是实函数。



如果以特征向量 $\Phi(x)$ 代替输入向量 x , 则由(8.28)式和(8.31)可以得到:

$$D_{ij} = y_i y_j \Phi(x_i) \bullet \Phi(x_j) \quad (8.33)$$

$$d(x) = \Phi(x) \bullet w^* + b^* = \sum_{i=1}^l \alpha_i y_i \Phi(x_i) \bullet \Phi(x) + b^* \quad (8.34)$$

由上可知,不论是寻优函数(8.26)式还是分类函数(8.31)都只涉及训练样本之间的内积 $(x_i \bullet x_j)$ 。这样,在高维空间实际上只需进行内积运算,而这种内积运算是可以用原空间中的函数实现的,我们甚至没有必要知道变换的形式。根据泛函的有关理论,只要一种核函数 $K(x_i \bullet x_j)$ 满足 Mercer 条件,它就对应某一空间中的内积[Vapnik 95]。

因此,在最优分类面中采用适当的内积函数 $K(x_i \bullet x_j)$ 就可以实现某一非线性变换后的线性分类,而计算复杂度却没有增加。此时的目标函数(8.26)式变为

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j y_i y_j K(x_i, x_j) \quad (8.35)$$

而相应的分类函数也变为

$$d(x) = \sum_{i=1}^l y_i \alpha_i^* K(x, x_i) + b^* \quad (8.36)$$

这就是支持向量机。

对给定的 $K(x,y)$, 存在对应的 $\Phi(x)$ 的充要条件是: 对于任意给定的函数 $g(x)$, 当 $\int_a^b g(x)^2 dx$ 有限时,有

$$\int_a^b \int_a^b K(x,y) g(x) g(y) dx dy \geq 0 \quad (8.37)$$

这个判别条件不容易实际操作,有一个简单的方法: 因为多项式是满足 Mercer 条件的,因此只要 $K(x,y)$ 能被多项式逼近,就满足 Mercer 条件。

构造类型判定函数的学习机称为支持向量机,在支持向量机中构造的复杂性取决于支持向量的数目,而不是特征空间的维数。支持向量机的示意图如图 8.6 所示。

对非线性问题支持向量机首先通过用内积函数定义的非线性变换将输入空间变换到一个高维空间,在这个空间中求广义最优分类面。

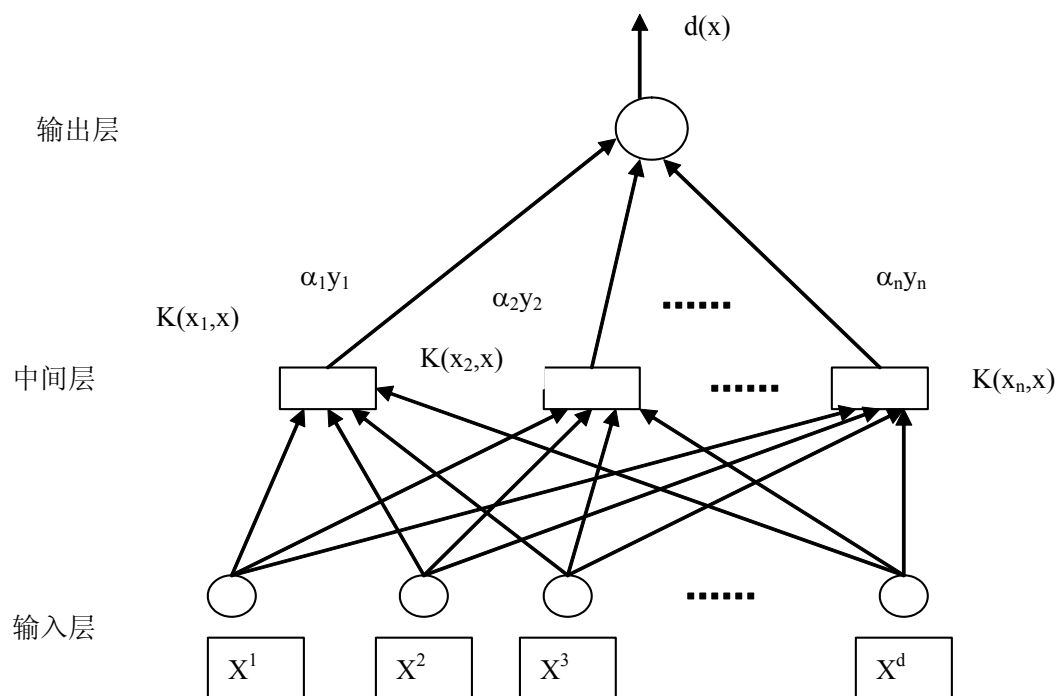


图 8.6 支持向量机的示意图

8.5 核函数

支持向量机中不同的内积核函数将形成不同的算法。目前常用的核函数主要有多项式核函数、径向基函数、多层感知机、动态核函数等。

8.5.1 多项式核函数

多项式核函数:

$$K(x, x_i) = [(x, x_i) + 1]^d \quad (8.38)$$

所得到的是 d 阶多项式分类器。

$$f(x, \alpha) = \text{sign} \left(\sum_{\substack{\text{sup port} \\ \text{vector}}} y_i \alpha_i [(x_i \bullet x) + 1]^d - b \right)$$

8.5.2 径向基函数

经典的径向基函数使用下面的判定规则::

$$f(x) = \text{sign} \left(\sum_{i=1}^l \alpha_i K_\gamma(|x - x_i|) - b \right) \quad (8.39)$$

其中, $K_\gamma(|x - x_i|)$ 取决于两个向量之间的距离 $|x - x_i|$ 。对于任何 γ 值, 函数 $K_\gamma(|x - x_i|)$ 是一个非负的单调函数。当训练样本数趋向无穷大时它趋向零。最通用的判定规则是采用高斯函数:

$$K_\gamma(|x - x_i|) = \exp \left\{ -\frac{|x - x_i|^2}{\sigma^2} \right\} \quad (8.40)$$

构造(8.39)式的判定规则, 必须估计:

- (1) 参数 γ 的值;
- (2) 中心点 x_i 数目 N ;
- (3) 描述中心点向量 x_i ;
- (4) 参数 α_i 的值。

与传统的径向基函数方法的重要区别是, 这里每个基函数的中心点对应一个支持下向量, 中心点本身和输出权值都是由 SVM 训练算法来自动确定的。

8.5.3 多层感知机

支持向量机采用 Sigmoid 函数作为内积, 这时就实现了包含一个隐层的多层感知机。隐层节点数目由算法自动确定。满足 Mercer 条件的 Sigmoid 核函数为

$$K(x_i, x_j) = \tanh(\gamma x_i^T x_j - \Theta) \quad (8.41)$$

算法不存在困扰神经网络的局部极小问题。

8.5.4 动态核函数

1999 年 Amari 和 Wu 通过对核函数的黎曼几何分析, 提出利用实验数据逐步修正原有的核函数, 使之更好适应实际问题。设特征映射 $U = \Phi(x)$, 则

$$dU = \sum_i \frac{\partial}{\partial x_i} \Phi(x) dx_i$$

$$\|dU\|^2 = \sum_{i,j} g_{ij}(x) dx_i dx_j$$

这里 $g_{ij}(x) = \left(\frac{\partial}{\partial x_i} \Phi(x) \right) \cdot \left(\frac{\partial}{\partial x_j} \Phi(x) \right)$ 称非负定阵 $(g_{ij}(x))$ 为 R^n 上的黎曼张量,

$ds^2 = \sum_{ij} g_{ij}(x) dx_i dx_j$ 为 R^n 上的黎曼距离。赋予黎曼距离的 R^n 成为黎曼空间, 体积

$$dv = \sqrt{g(x)} dx_1 \cdots dx_n$$

其中, $g(x) = \det(g_{ij}(x))$ 。直观地说, $g(x)$ 反映了特征空间中, 点 $\Phi(x)$ 附近局部区域被放大的程度。因此, 也称 $g(x)$ 为放大因子。

因为 $k(x, z) = (\Phi(x) \bullet \Phi(z))$, 可以验证

$$g_{ij}(x) = \frac{\partial}{\partial x_i \partial z_j} k(x, z) \Big|_{z=x}$$

特别对高斯函数 $k(x, z) = \exp\left\{-\frac{\|x-z\|^2}{2\sigma^2}\right\}$, $g_{ij}(x) = \frac{1}{\sigma^2} \delta_{ij}$ 。

为了有效地将两类不同的模式区分开, 希望尽量拉大它们之间的距离, 即尽量放大分离曲面附近的局部区域。可以用修正核函数的办法达到此目的。设 $c(x)$ 是正的可微实函数, $k(x, z)$ 是高斯核, 则

$$\tilde{k}(x, z) = c(x)k(x, z)c(z) \quad (8.42)$$

也是核函数, 且

$$\tilde{g}_{ij}(x) = c_i(x)c_j(x) + c^2(x)g_{ij}$$

这里, $c_i(x) = \frac{\partial}{\partial x_i} c(x)$ 。Amar i 和 Wu 设 $c(x)$ 有如下的形式

$$c(x) = \sum_{x_i \in SV} h_i e^{-\frac{\|x-x_i\|^2}{2\tau^2}} \quad (8.43)$$

这里, $\tau > 0$ 是参数, h_i 是权系数。在支持向量 x_i 附近, 有

$$\sqrt{\tilde{g}(x)} \approx \frac{h_i}{\sigma^n} e^{-\frac{nr^2}{2\tau^2}} \sqrt{1 + \frac{\sigma^2}{\tau^4} \gamma^2}$$

其中, $\tau = \|x - x_i\|$ 是欧几里德距离, 为保证 $\sqrt{\tilde{g}(x)}$ 在 x_i 附近取最大值, 同时在其它区域取较小值。经计算知

$$\tau \approx \frac{\sigma}{\sqrt{n}} \quad (8.44)$$

这样, 新的训练过程由两步组成:

- (1) 先用某个核 k (高斯核) 进行训练, 然后按照 (8.42)、(8.43)、(8.44) 得到修正的核 \tilde{k} ;
- (2) 用 \tilde{k} 进行训练。

这种改进的训练方法, 不仅可以明显地降低错误识别率, 而且, 还可减少支持下向量的个数, 从而提高识别速度。

8.6 基于分类超曲面的海量数据分类方法

张铃与张钊教授采用球面投影函数作为非线性映射,完成样本点的分类问题,这与 Vapnik 的思想在本质上是相同的[Zhang Lin 1999],其算法可以说明为:在球面上,考虑一个被映射到球面上的样本,并建立以其坐标为中心的邻域,使得在这个邻域中最大限度地包括相同类别的样本。显然,这个邻域可以通过在原空间中的距离(样本之间的内积)来定义。然后,将这些样本删除,再重复上述过程,直到覆盖所有的样本。与 Vapnik 的支持向量机相比较,他们未显现地使用向量这个概念,而使用邻域来代替,而两种方法的区别就在求解这个二次规划问题的方法上:Vapnik 直接计算(8.45)式,

$$Q(\alpha_i) = \sum \alpha_i - \sum \alpha_i \alpha_j y_i y_j K(x_i \bullet x_j) \quad (8.45)$$

这需要计算出一个 m 维的内积矩阵,而基于邻域的方法则是在计算样本之间内积的同时,边判断哪些样本可以删除,即可以使用邻域概括一部分 $\alpha_i = 0$ 的向量,因为每删除一个样本就意味着使得基于(8.45)式的内积矩阵降低一维,因此,这个考虑可以大大减少所需要的存储空间与计算量。在应用上,它使得基于 SVM 原理处理海量样本集成为可能,这是对统计机器学习理论的重要贡献。

8.6.1 Jordan 曲线定理

设 $X \subset R^2$ 是平面的闭子集, X 同胚与圆周,那么它的余集 $R^2 \setminus X$ 有两个连通分支,一个是有限界的,另一个是无界的, X 中任何一点的任何邻域与这两个连通分支均相交。

推论 8.2 设 D 是一个闭圆盘, D^0 为其内部, C 为其边界圆周, $f: D \rightarrow R^2$ 是一个连续的一一映射,那么 $R^2 \setminus f(C)$ 有两个连通的分支 $f(D^0)$ 和 $R^2 \setminus f(D)$,特别地 $f(D^0)$ 是平面的一个开子集。

Jordan 曲线定理表明任何的简单闭曲线(一条两端相接并且自身不相交的曲线)都把一个平面分成两个区域——一个外部和一个内部。给定一个点,如何判断它在曲线的内部还是在曲线的外部呢?

定义 8.4 围绕数 对于任何连续曲线 $\gamma: [a, b] \rightarrow R^2 \setminus \{P\}$, 我们定义围绕数 $W(\gamma, P)$ 如下:

1. 将小区间划分为 $a = t_0 \leq t_1 \leq \dots \leq t_n = b$, 使得每个小区间 $[t_{i-1}, t_i]$ 映射到顶点为 P 的小曲线段, 根据 Lebesgue 引理这样的划分是存在的, 因为 γ 的每个象点都包含于这样的小曲线段。

2. 选取包含 $\gamma([t_{i-1}, t_i])$ 的小曲线段 U_i 及其对应的极角函数 θ_i ($1 \leq i \leq n$)。设 $P_i = \gamma(t_i)$, $0 \leq i \leq n$ 。定义

$$W(\gamma, P) = \frac{1}{2\pi} \sum_{i=1}^n [(\theta_i(P_i) - \theta_i(P_{i-1}))] \quad (8.46)$$

命题 8.1 围绕数与步骤 1 与步骤 2 中的选取方法无关，若 γ 为闭曲线，即 $\gamma(a) = \gamma(b)$ ，则 $W(\gamma, P)$ 为一个整数。

定理 8.4 分类判别：设 $X \subset R^2$ 是平面的闭子集， X 同胚与圆周，那么它的余集 $R^2 \setminus X$ 有两个连通分支，一个是内部，另一个是外部，任取 $x \in R^2 \setminus X$ ，则

$x \in X$ 的内部 $\Leftrightarrow W(X, x)$ 为奇数， $x \in X$ 的外部 $\Leftrightarrow W(X, x)$ 为偶数。

上述定义与定理可推广到高维空间。

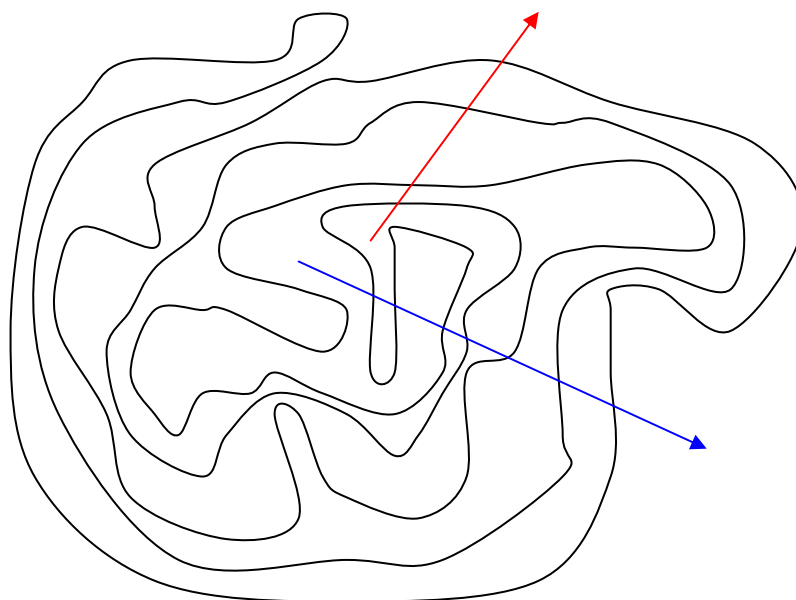


图 8.7 分类判别定理图示

基于拓扑学中的 Jordan 曲线定理，不需要考虑使用何种核函数，而通过使与球面同胚的双侧闭曲面作为分类超曲面（Separating Hyper Surface）对空间进行划分。分类超曲面可以由多个超平面构成，而点属于超曲面内部还是外部取决于该点引出的射线与超曲面相交为奇数还是偶数，该判别方法使得基于非凸的超曲面的分类判别变得直接、简便。

8.6.2 SVM 直接方法基本思想

根据上述定理我们提出如下 SVM 直接方法的基本思想用于二元数据分类，下述方法有望推广到高维数据的处理。

1. 设已给的样本点落在一个矩形区域中；
2. 将矩形区域划分成若干小区域，使每个小区域至多含一个样本点；

3. 对每个含样本点的小区域边界根据样本点的类别标定方向，一种为顺时针方向，一种为逆时针方向；
4. 合并矩形边界，同向线段保留，异向线段抵消，获得若干封闭折线组成的分类超平面，并以链的形式存储超平面；
5. 输入新样本点，计算该点关于以上分类曲线的围绕数，根据围绕数判定该样本点所在的类；另一种简便方法是选择适当的由待定点出发的射线，通过射线与超平面的相交数的奇偶性判断样本点所在的类；若不能判断，就围绕该点做一个小矩形，并对边界定向，之后转入第 4 步；
6. 若是多类判断问题，就对第 3 步的小矩形的向量增加一个标类别的分量，第 4 步用同类合并的办法。

8.6.3 实现算法

根据上述基本思想，设计出 SVM 直接方法的训练算法及分类算法（二元）如下。

1. 学习过程

设矩形区域为归一化单元格（如图 8.8），训练样本落入此区域内一单元格内，按照类别（X、O）将此单元格分为两类，分别标注边界；将同类区域边界合并，并以链表形式存放。

2. 细化方案

若单元格内已有一不同类训练样本，则将此单元格细化，并进行归一化操作，继续标注，再合并边界，存放边界链表；循环完成此训练过程。

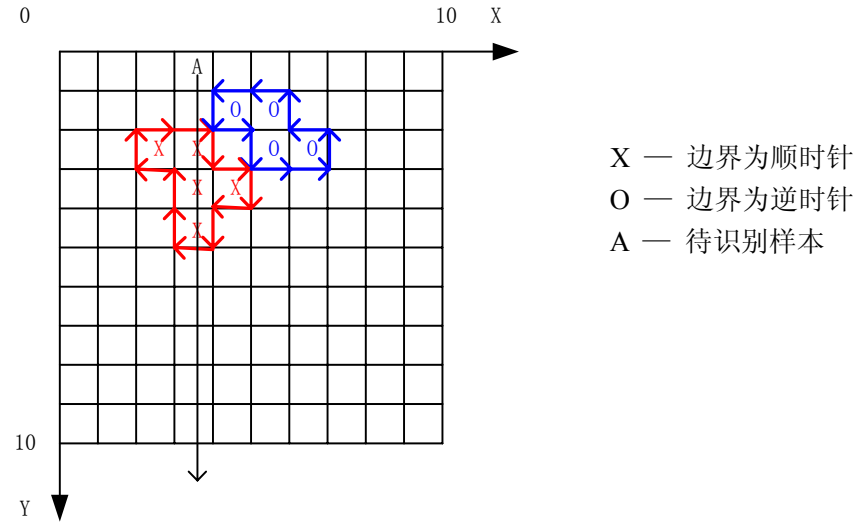


图 8.8 实现方案

3. 分类过程

当一待识别的样本（A）进入此区域后，作射线，与存储链表形成的区域多边形各边作相交操作，根据判别点是否在多边形内的规则，即两类边界围绕数的奇偶性，判断出此样本所属类别。

如果样本所在小区域已经细化，则将样本坐标单位化，放入细化区域，继续进行上述的

分类过程。

根据分类超曲面的思想，我们给出以上几步算法实现过程，训练过程与分类过程的算法复杂度都是多项式的。实际细化过程中，为保证边界的连续性，链表也要同时记录同区域内同类训练样本，但在样本集规模很大的情况下，可对细化的层次进行控制，即在训练过程中将不影响分类边界生成的样本删除，可保证计算速度。

8.6.4 实验结果分析

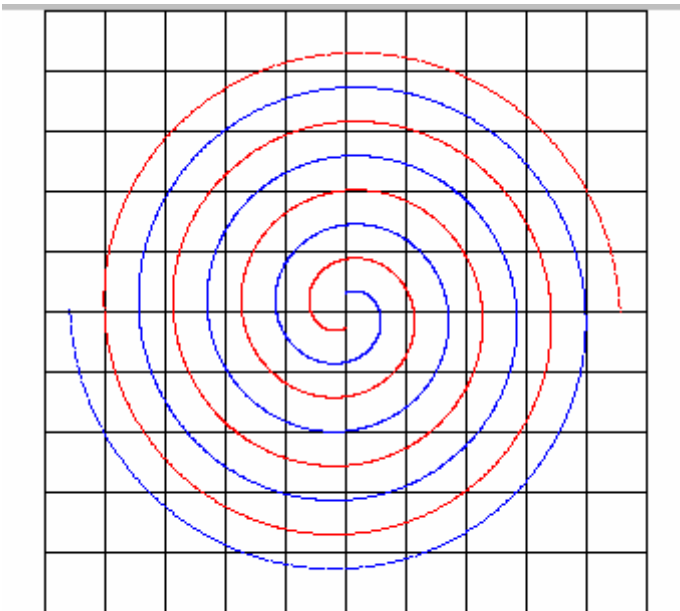
我们已经提出了基于分类超曲面的分类算法，本节使用 Spiral 螺线两类样本数据（线形不可分），测试本算法的分类正确率。可以看到随着样本数的增加，形成的分类折线连贯性越好，分类测试准确率越高。对于海量数据（ 10^7 ），SVM 直接方法的计算速度极快，同时对计算机资源要求很低，而传统的 SVM 不具备有这种优点，由此可见，对大规模样本点进行分类，SVM 直接方法可得到良好的结果。

1.构造测试数据

双螺旋分类问题：两条螺旋线 K_1 和 K_2 （极坐标形式）

$$\begin{aligned} K_1: \rho &= \theta \\ K_2: \rho &= \theta + \pi \end{aligned} \quad \frac{\pi}{2} \leq \rho \leq 8\pi \quad (8.47)$$

根据公式（8.47）构造训练样本集合及测试样本集合，圈数=4，不失一般性，将样本点平移并压缩至算法所需范围内，如图 8.9 所示。



Samples Num: 5402

图 8.9 双螺旋问题

实验经过以下几个步骤：

- (1) 生成训练样本，导入数据库；
- (2) 对训练样本进行训练，记录训练所需时间；保存训练所得分类链表；
- (3) 从数据库提取分类链表，对测试样本进行分类，记录测试所需时间，记录分类结果，计算分类正确率。测试样本的选取一类为训练样本本身集合，一类为数量多倍于训练样本的另一样本集合。

1. 中小规模样本实验结果

(1) 训练结果见表 8.1。

表 8.1 中小规模样本训练结果

样本点个数①	训练所需时间②	分类所需时间③	分类正确率 (%) ④
5, 402	2s	4s	100.00
13, 500	6s	10s	100.00
27, 002	13s	21s	100.00
54, 002	27s	43s	100.00
108, 000	57s	1m 24s	100.00
540, 000	6m 21s	8m 11s	100.00
1, 350, 002	19m 14s	20m 10s	100.00
5, 400, 000	46m 45s	1h 10m 5s	100.00

①样本点为 Spiral 螺线等角速度构造，②时间表示为间隔：h（小时）、m（分钟）、s（秒），③测试样本点集合为训练样本点本身集合，④测试样本点实际类别为构造时所得。

(2) 训练所得分类链表。

训练所得分类链表的保存和提取均可在几秒内完成，且需要的存储空间极少。训练所得分类链表结构见图 8.10。

2. 大规模样本实验结果

(1) 训练结果见表 8.3。

表 8.3 大规模样本训练结果

样本点个数	训练所需时间	分类所需时间①	分类正确率 (%)
10, 800, 000	1h 34m 57s	2h 17m 35s	100.00
22, 500, 002	3h 16m 9s	4h 49m 55s	100.00
54, 000, 000	7h 42m 52s	11h 47m 7s	100.00

①测试样本点集合为训练样本点本身集合。

(2) 分类测试结果见表 8.4。

表 8.4 大规模样本测试结果

样本点个数①	测试样本点个数	分类所需时间	分类正确率 (%)
10, 800, 000	22, 500, 002	4h 7m 4s	100.00
22, 500, 002	54, 000, 000	11h 25m 3s	100.00
54, 000, 000	67, 500, 002	14h 37m 6s	100.00

①样本点个数为训练点个数

3. 小样本训练，大样本分类测试结果见表 8.5。

表 8.5 小样本训练，大样本分类测试结果

样本点个数①	测试样本点个数②	分类所需时间	分类正确率 (%)
5, 402	54, 002	41s	99.59
5, 402	540, 000	6m 45s	99.58
27, 002	540, 000	6m 44s	99.98
54, 002	540, 000	6m 47s	100.00
54, 002	5, 400, 000	1h 7m 7s	100.00

①样本点个数为训练点个数，②测试样本点由 Spiral 螺线构造的另一样本集合（样本数量为训练样本的 10 倍以上）。

表 8.5 表明基于分类超曲面的 SVM 直接方法有很好的泛化能力。

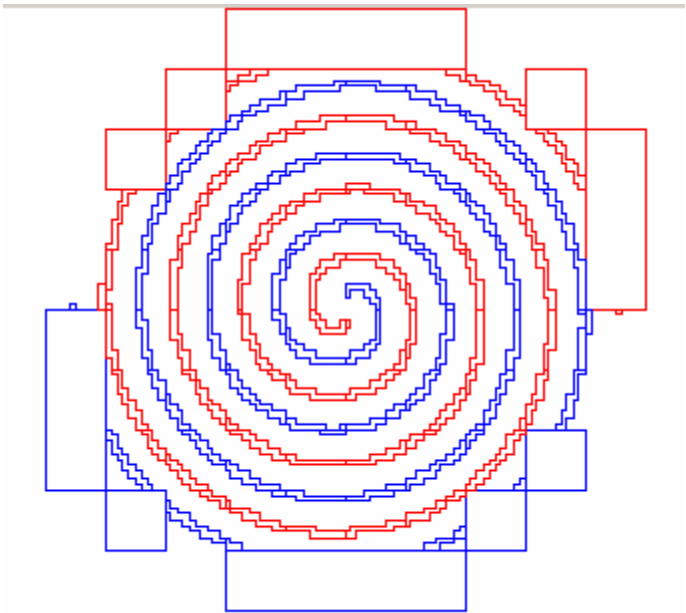
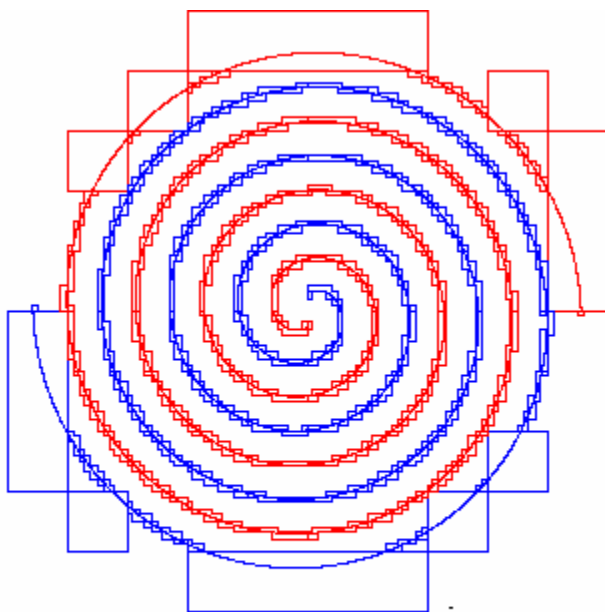


图 8.10 训练所得分类链表结构

训练所得分类链表及其对样本点的覆盖情况见图 8.11。



样本数:27002
正确率: 100000%

图 8.12 分类链表及其对样本点的覆盖情况

习题

1. 比较经验风险最小化原理和结构风险最小化原理。
2. VC 维的含义是什么，为什么说 VC 维反映了函数集的学习能力。
3. 试叙述统计学习理论的三个里程碑，并分析各个里程碑具体解决了什么问题。
4. 描述支持向量机的基本思想和数学模型。
5. 为什么说统计学习理论是支持向量机的理论基础，表现在哪些方面。
6. 考虑用于线性可分模式的超平面，它的方程定义为：

$$w^T x + b = 0$$

其中 w 表示权值向量， b 为偏置， x 为输入向量。如果输入模式集 $\{x_i\}_{i=1}^N$ 满足附加的条件

$$\min_{i=1,2,\dots,N} |w^T x_i + b| = 1$$

则称 (w, b) 为超平面的标准对(canonical pair)。证明标准对的要求导致两类分离边界的距离为 $2/\|w\|$ 。

7. 简述支持向量机解决非线性可分问题的基本思想。
8. 两层感知器的内积核定义为

$$K(x, x_i) = \tanh(\beta_0 x^T x_i + \beta_1)$$

探讨对常数 β_0 和 β_1 的那些值不满足 Mercer 定理的条件

9. 关于下列任务比较支持向量机和径向基函数(RBF)网络的优点和局限:

- (1) 模式识别
- (2) 非线性回归

10. 比较其它的分类方法, 讨论支持向量机的几个重要优势, 并从理论上的加以解释。

第九章 解释学习

解释学习 (Explanation-Based Learning, 简称 EBL) 是一种分析学习方法, 在领域知识指导下, 通过对单个问题求解实例的分析, 构造出求解过程的因果解释结构, 并获取控制知识, 以便用于指导以后求解类似问题。

9.1 概述

解释学习最初是由美国 Illinois 大学的 DeJong 于 1983 年提出来的。在经验学习的基础上, 运用领域知识对单个例子的问题求解作出解释, 这是一种关于知识间因果关系的推理分析, 可产生一般的控制策略。

1986 年, Mitchell, Keller 和 Kedar-Cabelli 提出了解释的泛化 (Explanation-Based Generalization, 简称 EBG) 的统一框架, 把解释学习的过程定义为两个步骤:

- (1) 通过分析一个求解实例来产生解释结构;
- (2) 对该解释结构进行泛化, 获取一般的控制规则。

DeJong 和 Mooney 提出了更一般的术语——解释学习 (Explanation-based Learning, 简称 EBL)。从此解释学习成为机器学习中的一个独立分枝。基于解释的学习从本质上说属于演绎学习, 它是根据给定的领域知识, 进行保真的演绎推理, 存储有用结论, 经过知识的求精和编辑, 产生适合以后求解类似问题的控制知识。解释学习与经验学习的方法不同, 经验学习是对大量训练例的异同进行分析, 而解释学习则是对单个训练例 (通常是正例) 进行深入的、知识集约型的分析。分析包括: 首先解释训练例为何是欲学概念 (目标概念) 的一个例子, 然后将解释结构泛化, 使得它能比最初例子适用于更大一类例子的情况, 最后从解释结构得到更大一类例子的描述, 得到的这个描述是最初例子泛化的一般描述。

解释学习可以用于获取控制知识、精化知识、软件重用、计算机辅助设计和计算机辅助教育等方面。在传统程序中解释的作用主要是说明程序、给定提示、向用户提供良好的可读性。按人工程序的特点, 解释已被赋予新的含义, 其作用是:

- (1) 对所产生的结论的推理过程作详细说明, 以增加系统的可接受性;
- (2) 对错误决策进行追踪, 发现知识库中知识的缺陷和错误的概念;
- (3) 对初学的用户进行训练。

解释的方法也因此由简单变得复杂了。一般采用的解释方法有:

- (1) 预制文本法。预先用英文写好, 并插入程序中;

(2) 执行追踪法。遍历目标树, 通过总结与结论相关的目标, 检索相关规则, 以说明结论是如何得到的;

(3) 策略解释法。明确表示控制知识, 即用元知识概括地描述, 与领域规则完全分开。从策略的概括表示中产生解释, 能为用户提供问题求解策略的解释。

在解释学习中主要采用了执行追踪法。通过遍历目标树, 对知识相互之间的因果关系给出解释, 而通过这种因果关系的分析, 学习控制知识。

9.2 解释学习模型

R. M. Keller 指出解释学习涉及三个不同的空间：例子空间、概念空间和概念描述空间 [Keller 1987]。一个概念可由例子空间外延地表示成某些事例的集合。概念也可以由概念描述空间内涵地表示为例子空间例子的属性。图 9.1 说明了三个空间的关系。

概念空间指某个学习程序能描述的所有概念的集合，其中每个点对应例子空间的唯一的一个子集合。例如 C_1 对应 I_1, I_2, I_3 。但是概念空间中的一个点可以对应概念描述空间的多个点，这些点分成可操作的和不可操作的两部分。例如 C_1 对应 D_1 (不可操作的) 和 D_2 (可操作的)。对应同一概念的两个描述称为同义词。如 D_1 和 D_2 是同义词。解释学习的任务就是由不可操作的描述转化为可操作的描述。

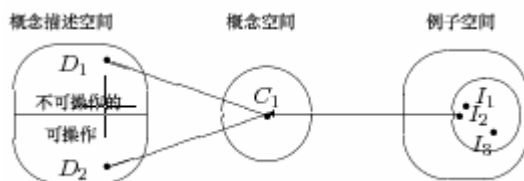


图 9.1: 解释学习的空间描述

在解释学习中，如图 9.1 所示， D_1 是提给系统初始的、不可操作的描述。而 D_2 是学习到的最终的、可操作的描述。所以 D_1 可看成是搜索的开始结点， D_2 是解结点，解释是空间变换，而可操作性是搜索结束的标准。从 D_1 到 D_2 变换的过程称作概念可操作 [Keller 1987]。

从概念上讲，每种解释学习系统都包括可操作性评估过程，评价概念的描述，产生可操作性评测结果。可变性、粒度、确定性是可操作性在评估过程中产生的三维特性。表 9.1 给出了几个系统的可操作特性。

表 9.1: 可操作特性

系统	可变性	粒度	确定性
GENESIS	动态	二进制	不保证
LEX2	动态	二进制	不保证
SOAR	动态	二进制	不保证
PRODIGY	静态	连续	不保证
MetaLEX	动态	连续	保证

根据上述的空间描述，可以建立解释学习的一种模型。解释学习系统主要包括执行系统 PS，学习系统 EXL，领域知识库 KB (这是不同描述间转换规则的集合)。令概念空间为 C ，概念描述空间为 CD ，例子空间为 I 。则系统工作过程如下。EXL 的输入是概念 C_1 的描述 D_1 (一般是不可操作的)。根据 KB 中的知识，对 D_1 进行不同描述的转换 (这是一个搜索过程)。PS 对每个转换结果进行测试，直至转换结果是 PS 可接受的描述 D_2 (是可操作的) 时，学习结束并输出 D_2 模型如图 9.2 所示。

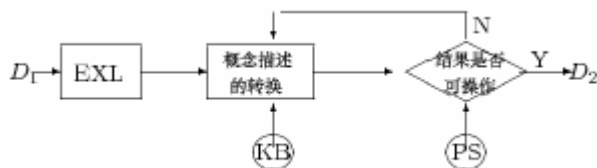


图 9.2: 解释学习的模型

9.3 解释泛化学习方法

9.3.1 基本原理

Mitchell, Keller 和 Kedar-Cabelli 于 1986 年为解释泛化学习方法提出了一个统一的框架 [Mitchell 1986]。其基本思想是对某一情况建立一解释结构, 将此解释结构概括使之可应用于更广泛的情况。解释泛化学习运用了知识的逻辑表示和演绎型问题求解方法。

为了便于叙述, 我们先引入一些术语。概念是例子空间上的谓词, 表示例子空间的一些子集。概念由其属性和属性值表示。概念定义说明了作为概念的一个例子应满足的充分必要条件, 而充分概念定义说明的只是充分条件。满足概念定义的例子称为实例或正例, 而不满足概念定义的例子则称为反例。一个实例的概括是描述包含此实例的例子集合的一个概念定义, 解释结构是证明树, 其中每个已例示的规则均被对应的泛化规则代替。

解释泛化学习问题可以形式化地描述为:

已知:

- 目标概念 (Goal concept):
要学习概念的描述。
- 训练例 (Training example):
目标概念的一个实例。
- 领域理论 (Domain theory):
用于解释训练例的一组规则、事实。
- 可操作性标准 (Operationality criterion):
说明概念描述应具有的形式谓词。

欲求:

- 训练实例的泛化, 使之满足以下两个条件:
 - (1) 是目标概念的充分概念描述。
 - (2) 满足可操作性标准。

解释泛化学习方法可以分成两个阶段。第一阶段是解释, 这个阶段的任务就是利用领域理论中的知识, 对训练实例进行解释, 建立解释树, 或叫证明树, 证明训练例如何满足目标概念定义。树的叶结点满足可操作性标准。第二阶段的工作是泛化, 也就是对第一阶段的结果——证明树进行处理, 对目标概念进行回归, 从而得到所期望的概念描述。实际上, 第一阶段解释的工作是将实例的相关属性与无关属性分离开来, 第二阶段泛化的工作是分析解释结构。

(1) 产生解释。用户输入实例后, 系统首先进行问题求解。如由目标引导反向推理, 从领域知识库中寻找有关规则, 使其后件与目标匹配。找到这样的规则后, 就把目标作为后件, 该规则作为前件, 并记录这一因果关系。然后以规则的前件作为子目标, 进一步分解推理。如此反复, 沿着因果链, 直到求解结束。一旦得到解, 便证明了该例的目标可满足, 并获得了证明的因果解释结构。

构造解释结构通常有两种方式: 一是将问题求解的每一步推理所用的算子汇集, 构成动作序列作为解释结构; 另一种是自顶向下的遍历证明树结构。前者比较概括, 略去了关于实例的某些事实描述; 后者比较细致, 每个事实都出现在证明树中。解释的构造可以在音量求解的同时进行, 也可在问题求解结束后, 沿着解路径进行。这两种方式形成了边解边学和解完再学两种方法。

(2) 对得到的解释结构核心事件进行泛化。在这一步，通常采取的办法是将常量转换为变量，即把例子中的某些具体数据换成变量，并略去某些不重要的信息，只保留求解所必需的那些关键信息，经过某种方式的组合，形成产生式规则，从而获得泛化的控制知识。

作为一个例子，考虑目标概念 $\text{SAFE-TO-STACK}(x, y)$ ，即一对物体 $\langle x, y \rangle$ ， x 可以安全地放在 y 的上面。 SAFE-TO-STACK 的实例如下：

设定：

- 目标概念：一对物体 $\langle x, y \rangle$ ， $\text{SAFE-TO-STACK}(x, y)$ ，其中 $\text{SAFE-TO-STACK}(x, y) \Leftrightarrow \text{NOT}(\text{FRAGILE}(y)) \vee \text{LIGHTER}(x, y)$ 。

- 训练实例：

ON(OBJ1, OBJ2)
ISA(OBJ1, BOX)
ISA(OBJ2, ENDTABLE)
COLOR(OBJ1, RED)
COLOR(OBJ2, BLUE)
VOLUME(OBJ1, 1)
DENSITY(OBJ1, .1)
...

- 领域知识：

$\text{VOLUME}(p1, v1) \wedge \text{DENSITY}(p1, d1) \rightarrow \text{WEIGHT}(p1, v1*d1)$
 $\text{WEIGHT}(p1, w1) \wedge \text{WEIGHT}(p2, w2) \wedge \text{LESS}(w1, w2) \rightarrow \text{LIGHTER}(p1, p2)$
 $\text{ISA}(p1, \text{ENDTABLE}) \rightarrow \text{WEIGHT}(p1, 5) (\text{default})$
 $\text{LESS}(.1, 5)$
...

- 可操作标准：概念定义必须要用描述实例中的谓词(例如 VOLUME, COLOR, DENSITY)，或者选自领域知识易于评测的谓词(例如 LESS)。

确定：

- 训练实例的泛化是对目标概念给以充分定义，并且满足可操作性标准。

首先 EBG 系统利用领域知识建造解释树，使训练实例满足目标概念的定义。图 9.3 给出 $\text{SAFE-TO-STACK}(\text{OBJ1}, \text{OBJ2})$ 的解释树。

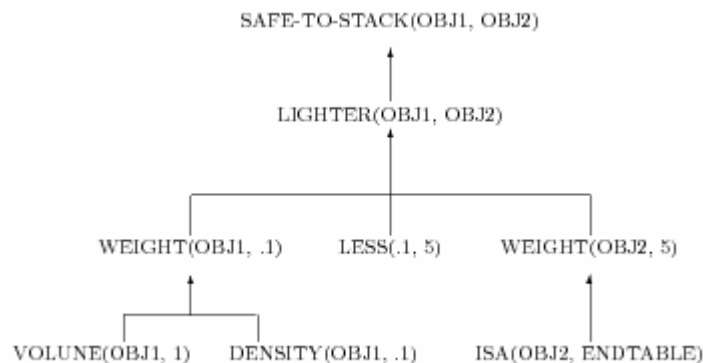


图 9.3 $\text{SAFE-TO-STACK}(\text{OBJ1}, \text{OBJ2})$ 解释树

图 9.4 给出了 SAFE-TO-STACK 实例处理的泛化步骤。根据领域知识提供的规则 $\text{LIGHT}(p1, p2) \rightarrow \text{SAFE-TO-STACK}(p1, p2)$ ，目标概念表达式 $\text{SAFE-TO-STACK}(x, y)$ 进行回归。同样， $\text{LIGHT}(x, y)$ 通过规则 $\text{WEIGHT}(p1, w1) \wedge \text{WEIGHT}(p2, w2) \wedge \text{LESS}(w1, w2) \rightarrow \text{LIGHT}(p1, p2)$

回归，产生的谓词显示在第二层。第三层 $WEIGHT(x, w1)$ 通过规则 $VOLUME(p1, v1) \wedge DENSITY(p1, d1) \rightarrow WEIGHT(p1, v1*d1)$ 回归，所以产生 $VOLUME(x, v1) \wedge DENSITY(x, d1)$ 。通过规则 $ISA(p2, ENDTABLE) \rightarrow WEIGHT(p2, 5)$ 回归 $WEIGHT(y, w2)$ 产生 $ISA(y, ENDTABLE)$ 。 $LESS(w1, w2)$ 被简单地加到结果表达式，因为没有规则可以合一。最后， $SAFE-TO-STACK(x, y)$ 的概念描述如下：

$VOLUME(x, v1) \wedge DENSITY(x, d1) \wedge LESS(v1*d1, 5) \wedge ISA(y, ENDTABLE) \rightarrow SAFE-TO-STACK(x, y)$

这个描述满足可操作性。

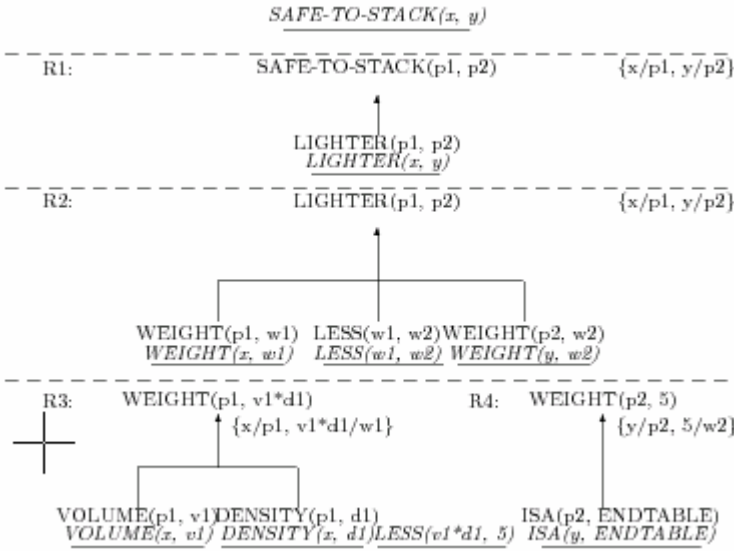


图 9.4 $SAFE-TO-STACK(OBJ1, OBJ2)$ 解释的泛化过程

解释学习实际上没有获得新知识，而只是把现有的不能用或不实用的知识转化为可用的形式。解释学习系统必须了解目标概念的初始描述，虽然该描述是正确无误的，但它是不可操作的。通俗地讲，这意味着学习程序不能有效地使用这个描述来提高性能，也就是说，概念描述本身与该种概念能否使用是不同的。解释学习系统的任务就是通过变换初始描述或使初始描述可操作化来缩小这种差距。

9.3.2 解释与泛化交替进行

1987 年斯坦福大学 Hirsh 提出了新的解释泛化方法[Hirsh 1987]，即将解释学习中的解释与泛化交替进行。

1. 问题的逻辑描述

学习系统提供了正向、反向、归纳等多种推理方式。逻辑的表示方法使 EBG 的语义更为清楚，为学习提供了方便的语言环境。例如要学习目标概念 Safe-to-Stack(V1, V2)，其领域知识就可表示如下：

事实知识：

On(obj1, obj2)
Isa(obj2, Endtable)
Color(obj1, red)
Color(obj2, blue)

Volme(obj1, 1)
 Density(obj1, 0.1)
 领域规则:
 $\text{Not}(\text{Fragile}(y)) \rightarrow \text{Safe-to-Stack}(x, y)$
 $\text{Lighter}(x, y) \rightarrow \text{Safe-to-Stack}(x, y)$
 $\text{Volume}(p1, v1) \wedge \text{Density}(p1, d1) \wedge X(v1, d1, w1)$
 $\rightarrow \text{Weight}(p1, w1)$
 $\text{Isa}(p1, \text{Endtable}) \rightarrow \text{Weight}(p1, 5)$
 $\text{Weight}(p1, w1) \wedge \text{Weight}(p2, w2) \wedge <(w1, w2)$
 $\rightarrow \text{Lighter}(p1, p2)$

2. 产生解释结构

为了证明该例子满足目标概念，系统从目标开始反向推理，根据知识库中已有的上述事实和规则，分解目标。每当使用一条规则时，同时返回去把该规则应用到变量化的目标概念上。这样，在生成该例子求解的解释结构的同时，也生成了变量化的泛化的解释结构。

3. 生成控制规则

将泛化的解释结构的所有叶结点的合取作为前件，以顶点的目标概念为后件，略去解释结构的中间部件，就生成泛化的产生式规则。使用生成的这个控制规则求解类似问题时，求解速度快且效率高。但是简单地把常量转为变量以实现泛化的方法可能过份一般化。在某些特例下可能使规则失败。

9.4 全局取代解释泛化方法

1986 年 DeJong 和 Mooney 提出全局取代解释泛化 (Explanation Generalization using Global Substitutions, 缩写 EGGS) 方法 [DeJong & Moonet 1986]。在 EBG 方法中，通过实例解释结构的目标概念回归，忽略析取实现泛化。DeJong 的 EGGS 方法通过解释构造单元构件，当解释合一时这些单元就连接起来。

机器人规划 STRIPS 是第一个进行泛化解释工作的 [Fikes, Hart and Nilsson, 1972]。STRIPS 是宏命令系统，构造和泛化机器人规划。当给定要达到的目标时，STRIPS 执行搜索，发现一个操作序列，将初始状态变换到目标状态。初始模型和目标如下：

初始世界模型：

INROOM(ROBOT, R_1)
 INROOM(BOX1, R_2)
 CONNECTS(D1, R_1 , R_2)
 CONNECTS(D1, R_2 , R_3)
 BOX(BOX_i)
 ⋮

$(\forall x, y, z) [\text{CONNECTS}(x, y, z) \Rightarrow \text{CONNECTS}(x, z, y)]$

目标公式：

$(\exists x) [\text{BOX}(x) \wedge \text{INROOM}(x, R1)]$

STRIPS 采用的操作符如下：

GOTHRU(D, r_1 , r_2)

```

Precondition: INROOM(ROBOT, r1) ∧ CONNECTS(D, r1, r2),
Delete list: INROOM(ROBOT, r1),
Add list: INROOM(ROBOT, r2)
PUSHTHRU(b, d, r1, r2)
Precondition: INROOM(ROBOT, r1) ∧ CONNECTS(D, r1, r2)
               ∧ INROOM(b, r1),
Delete list: INROOM(ROBOT, r1)
               INROOM(b, r1),
Add list: INROOM(ROBOT, r2)
               INROOM(b, r2)。

```

在 STRIPS 中泛化过程具体构造泛化的机器人规划，采用三角表表示，利用归结证明前题条件。图 9.5 给出了描述操作符 GOTHRU(d, r₁, r₂) 和 PUSHTHRU(b, d, r₁, r₂) 的三角表。

* INROOM(ROBOT,R1) * CONNECTS(D1,R1,R2)	GOTHRU(D1,R1,R2)	
* INROOM(BOX1,R2) * CONNECTS(D1,R1,R2) * INROOM(ROBOT,R2) * CONNECTS(x,y,z) ⇒ * CONNECTS(x,z,y)	PUSHTHRU(BOX1,D1,R2,R1)	
		INROOM(ROBOT,R1) INROOM(BOX1,R1))

图 9.5 三角表

三角表是有用的，它给出了操作符的前提条件怎样取决于其它操作符的影响和初始世界模型。STRIPS 的解释泛化算法如下：

for each equality between p_i and p_j in the explanation structure do

let θ be the MGU of p_i and p_j

for each pattern p_k in the explanation structure do

replace p_k with $p_k \theta$

STRIPS 使用具有三角表的解释泛化算法生成泛化操作符序列。第一步，对整个泛化表用变量代替常量。第二步，根据两条标准约束表：一个是保持操作符之间的依赖关系；另一个是泛化表中的操作符前提条件可被证明，这与原来规划中前提条件证明一样。通过泛化处理，STRIPS 产生泛化如图 9.6 所示的泛化表。

* INROOM(ROBOT,p2)		
* CONNECTS(p3,p2,p5)	GOTHRU(p3,p2,p5)	
* INROOM(p6,p5)		
* CONNECTS(p8,p9,p5)	* INROOM(ROBOT,p5)	PUSHTHRU(p6,p8,p5,p9)
* CONNECTS(x,y,z) \Rightarrow * CONNECTS(x,z,y)		
		INROOM(ROBOT,p9)
		INROOM(p6,p9))

图 9.6 泛化三角表

DeJong 等的学习方法是学习图式(schemata)。图式的核心思想是构造系统知识时, 要将为了达到一定目标的相关知识放在一个组。图式是一种操作符的偏序集, 简单的图式因果地连在一起。EGGS 采用动态的可操作性准则, 当系统学到了一个概念的一般化图式后, 该概念就可操作了。EGGS 的问题描述如下:

给定:

- 领域理论: 由三部分组成: ①领域中对象的种类和特性的定义; ② 一组关于对象特性和相互关系的推理规则; ③ 一组问题求解的操作和已知的泛化图式。
- 目标: 目标状态的一般规约。
- 初始世界状态: 关于世界及其特性的规约。
- 观察到的操作/状态序列(可选): 通过专家观察低级操作序列, 达到目标的示例而获得。有时, 这会漏掉一些操作, 必须从输入到目标状态进行推理。

确定:

- 达到目标状态的一个新模式。

EGGS 在泛化过程中始终维护两个独立的置换表 SPECIFIC 和 GENERAL。SPECIFIC 用于对解释结构的置换, 以得到对具体实例的解释, 而 GENERAL 置换得到泛化的解释。假设 σ 是 SPECIFIC 置换, γ 是 GENERAL 置换, 那么, 泛化过程的形式化描述如下:

```
for each equality between expression e1 and e2 in the explanation structure:
    if e1 is the antecedent of a domain rule and e2 is the consequent of a domain
    rule
```

```
then    let  $\emptyset$  = the most general unifier of  $e1\sigma$  and  $e2\sigma$ 
```

```
        let  $\sigma = \sigma \emptyset$  (* update SPECIFIC substitution)
```

```
        let  $\delta$  = the most general unifier of  $e1\gamma$  and  $e2\gamma$ 
```

```
        let  $\gamma = \gamma \delta$  (* update GENERAL substitution)
```

```
else let  $\emptyset$  = the most general unifier of  $e1\sigma$  and  $e2\sigma$ 
```

let $\sigma = \sigma \oslash$ (* update SPECIFIC substitution)

基于上述思想,Mooney 和 Bennett 提出了 EGGS 解释泛化算法,这种算法与抽象 STRIPS 算法非常相似:

算法 9.1 EGGS 解释泛化算法。

```

let  $\gamma$  be the null substitution {}.
for each equality between  $p_i$  and  $p_j$  in the explanation structure do:
    let  $p_i'$  be the result of applying  $\gamma$  to  $p_i$ .
    let  $p_j'$  be the result of applying  $\gamma$  to  $p_j$ .

    let  $\theta$  be the MGU of  $p_i'$  and  $p_j'$ .

let  $\gamma$  be  $\gamma \theta$ .

for each pattern  $p_k$  in the explanation structure do:
    replace  $p_k$  with  $p_k \gamma$ .

```

9.5 解释特化学习方法

1987 年卡耐基-梅隆大学的 Minton 和 Carbonell 提出解释特化 (Explanation-Based Specialization, 简写 EBS) 学习方法 [Minton 1987]。他们利用这种方法开发了学习系统 PRODIGY。它较好地克服了 EBG 方法中过份一般化的缺点。它从多种目标概念学习,其解释过程是对每个目标概念进行详细描述。解释过程结束后,把得到在有关目标概念的描述转换成一条相应的控制规则。可以用这些规则来选择合适的结点、子目标、算子及约束。具体方法如下。

PRODIGY 是一种与一些学习模块结合的通用问题求解器,它的体系结构如图 9.7 所示。基于操作符的问题求解器,对于推理规则和操作符的搜索提供一种统一的控制结构。问题求解器包括一个简单的理由维护子系统,规定有条件影响的操作符。问题求解器搜索可以通过控制规则导航。

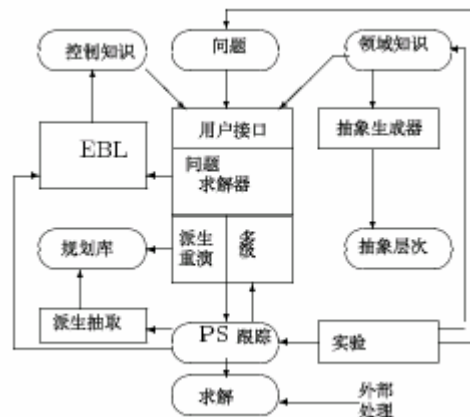


图 9.7 PRODIGY 体系结构

EBL 模块从问题求解跟踪获取控制规则。从领域和问题求解器的相关方面构造解释,然后结果表达式变成搜索控制规则。派生抽取模块是派生类比机。它可以重播过去类似问题的整个解。从图 9.7 可以看出,类比和 EBL 是独立的机制,获取具体领域的控制知识。实验学

习模块是为了求精不完全的或不正确的具体领域知识。当规划执行控制器检测到内部期望与外部期望不同时，实验模块就被触发。抽象产生器和抽象层次模块提供多级抽象规划能力。基于该领域的深度优先分析，领域知识被分成多个抽象级。当问题求解时 PRODIGY 生成抽象结果，并且进行求精。

PRODIGY 可以从四种目标概念学习，这些概念是：成功、失败、唯一的选择、目标互相制约。每当用户给定一个目标，以及它的一个例子，系统先反向分解目标到叶节点，得到相应的目标概念，然后分析问题求解轨迹，解释该例为何满足这个目标概念。

例如，如果得到了解，则将学习关于成功概念的控制规则；如果无解，则学习关于失败概念的控制规则；如果某个选择是唯一的，则学习有关唯一的选择概念的控制规则；如果某个目标的成功须依赖别的目标，则学习目标互相制约概念的控制规则。

1. 解释过程

PRODIGY 系统中的解释过程就是详细说明的过程。解释是使用训练例子提供的信息从知识库中寻找证明依据。解释过程等价于：以目标概念为根，生成一棵自顶向下的证明树。解释的每一步，都从领域知识库中选择与所给例子一致的规则，生成一个结点。每条规则是对结点子目标的详细描述。解释算法如下：

第一步：如果此概念是原语，则没有规则蕴含此概念，则不改变且返回，否则按第二步。

第二步：访问与此概念相联的识别器(即连接目标概念与知识库的映射函数)，取出与训练例子一致的规则。规则中每个非负的原子式为一个子概念。如果子概念没有被特化过，则特化该子概念；且重新唯一地命名已经特化过的变量；且用特化置换子概念并简化之。

第三步：返回。

这时的目标概念已是完全特化过的概念。用它们去套相应的规则模式，便获得一条对应此目标概念的控制规则。

2. 学习控制规则

在 PRODIGY 中，针对四种目标概念，有四种固定的控制规则模式。将某个目标概念的详细描述与规则模式匹配，就获得相应的控制规则。由成功概念学到 preference rules，它表明什么情况下某选择是成功的。由失败概念学到 rejection rules，它表明在这种情况下这种选择应该拒绝。如果其它选择都失败，则选择是唯一的，于是学到 selection rules。

下面以失败概念为例，说明解释过程以及控制规则的形成。领域背景积木世界的动作规划问题。对某个失败的规划动作的学习，产生的解释如下。

```
(OPERATOR-FAILS op goal node)if
(AND (MATCHES op (PICKUP x))
(MATCHES goal (HOLDING x))
(KNOWN node (NOT (ONTABLE x))))
```

其中小写字符是变量。上式是用领域知识库中形式化的知识对失败动作进行特化后的结果。其语义是，如果当前节点不是“ONTABLE x”，当前算子是“PICKUP x”，则算子“PICKUP x”是一个失败算子。由这个失败概念学到相应的 rejection rule 是：

```
(REJECT OPERATOR (PICKUP x)) if
(AND (CURRENT-NODE node)
(CURRENT-GOAL node (HOLDING x))
(CANDIDATE-OPERATOR (PICKUP x))
(OPERATE-FAILS op goal node))
```

其中 op=(PICKUP x)，goal=(HOLDING x)，node=(NOT (ONTABLE x))。规则含义是：如果当前节点是 (NOT (ONTABLE x))，当前目标是 (HOLDING x)，且该节点上候选算子是 (PICKUP x)，而且该算子在该节点和该目标是失败的，则拒绝算子 (PICKUP x)。

通过把控制规则传递到四个策略上，可以动态地改善问题求解器的性能。这四个策略是：选择节点、子目标、算子和一组约束的方法。在以后求解类似问题时，先用 selection rules 选出适合的子集，然后用 rejection rules 过滤，最后由 preference rules 寻找启发性最好的选择，从而达到减少搜索的效果。

3. 知识表示

PRODIGY 的知识库包括领域层公理和构筑层公理两类知识。前者是领域规则，后者是描述问题求解器的推理规则。二者都用陈述性逻辑语言表示，以便扩展和显式推理。

PRODIGY 虽然没有明显的泛化过程，但其领域规则，特别是问题求解器的推理规则实际上已经进行了泛化处理。它们不是由最原始的领域知识构成，所以实际上具有一定的概括性。另一方面，PRODIGY 是使用特化的方法由规则来构成证明树形式的解释，所以解释的结果所获取的描述是对目标概念的特化，对例子的泛化。这样的结果既保证不过份概括，又具有一定的通用性。

9.6 解释泛化的逻辑程序

如前所述，EBG 方法首先要建立一个关于训练例的解释结构。这实际上是定理证明问题，因此，可以把 EBG 看作是 Horn 子句归结定理证明的扩展，把泛化看作标准定理证明的附带工作。

9.6.1 工作原理

像 Prolog 之类的逻辑程序设计语言都是以 Robinson 提出的归结原理为基础的。归结原理的本质思想是检验短句集合是否包含空语句。实际上，归结原理是一种演绎规则，使用它能够从短句集中产生所有的结果短句。

归结原理：设两个短句 C_1 、 C_2 无公共变量， L_1 和 L_2 分别是 C_1 和 C_2 的两个文字，若 L_1 和 $\neg L_2$ 存在一个最一般的合一置换，那么子句 $(C_1 - L_1) \vee (C_2 - L_2)$ 就是两个子句 C_1 和 C_2 的归结式。

对于给定目标，Turbo Prolog 寻求匹配子句的详细过程就是合一过程，这个过程完成了其他程序设计语言中所谓的参量传递。下面是 Turbo Prolog 的合一算法。

算法 9.2 Turbo Prolog 的合一算法。

- (1) 自由变量可以和任意项合一。合一后该自由变量约束为与之合一的项。
- (2) 常量可与自身或自由变量合一。
- (3) 若两个复合项的函子相同且函子所带参量个数一样，则这两个复合项可以合一的条件是：所有子项能对应合一（表看成是特殊类型的复合项）。约束变量要用合一后的约束值替换。

EBG 的第一步工作——建立训练例的解释结构，就是以训练例为给定目标，通过搜索、匹配领域理论（领域理论已表示为 Prolog 的规则、事实），得到训练例的一个证明，证明它是目标概念的一个例子。既然如此，建立解释结构就需用到合一算法。基于这一点，把合一算法作为 EBG 算法实现的基础，在此基础上，进行 EBG 的第二步——泛化过程。

用 Turbo Prolog 实现 EBG，我们考虑把领域理论用内部数据库的形式存贮。这样在建立训练例的解释结构时，可以如同调用谓词一样，方便地查询数据库（领域知识），搜索匹配的

规则。

虽然 Prolog 的基础是合一算法，但用 Prolog 实现 EBG，仍需显式地执行合一算法，这就是说要用 Turbo Prolog 谓词完成合一算法，应该说，这实际上是元级程序设计的思想。具体地，我们定义谓词来处理项的合一、项表的合一等等。正是通过合一，完成了训练例的解释，建立一棵证明树。

第二步的泛化工作是用目标概念回归 (regress) 得到的证明树。这个回归过程包括常量用变量代替，以及新项合成等工作。用 Turbo Prolog 有效地实现常量用变量代替的过程是有困难的，尤其是当变量未被约束时。虽然已经定义了谓词实现回归，但如何增加其通用性，更有效地处理多个变量未被约束的情况，尚待进一步的工作去完成。

解释和泛化是 EBG 的两个阶段，但容易想到的是先解释，然后把得到的证明结构交给泛化阶段处理。这样不仅从时间效率上考虑，两者是串行工作的，而且分开进行，需要保存完整的证明树，各条路径都需保存。这里，我们考虑两步交叉进行。在系统试图证明训练例是目标概念的一个例子的同时，从目标概念出发，向后搜索，直到训练例的事实获得匹配为止。在这个过程中，每次运用一个规则，就同时倒回去，将此规则不经例示地用于已变量化了的的目标概念之上，从而在建立训练例的解释的过程中同时建立起泛化的解释结构。这样，就将 EBG 的两个阶段交叉起来。

9.6.2 元解释器

Prolog 简单的元解释器如下：

```
prolog(Leaf):-clause(Leaf, true).
prolog((Goal1, Goal2)):-
    prolog(Goal1),
    prolog(Goal2).
prolog(Goal):-
    clause(Goal, Clause),
    prolog(Clause).
```

在此基础上，使用 Prolog 构造元解释器作为 Prolog_EBG 的核心，具体程序如下：

```
prolog_ebg(X_Goal, X_Gen, [X_Goal], [X_Gen]):-clause(X_Goal, true).
prolog_ebg((X_Goal, Y_Goal), (X_Gen, Y_gen), Proof, GenProof):-
    prolog_ebg(X_Goal, X_Gen, X_Proof, X_GenProof),
    prolog_ebg(Y_Goal, Y_Gen, Y_Proof, Y_GenProof),
    concat(X_Proof, Y_Proof, Proof),
    concat(X_GenProof, Y_GenProof, GenProof).
prolog_ebg(X_Goal, X_Gen, [Proof], [GenProof]):-
    clause(X_Gen, Y_Gen),
    copy((X_Gen:-Y_Gen), (X_Goal:-Y_Goal)),
    prolog_ebg(Y_Goal, Y_Gen, Y_Proof, Y_GenProof),
    concat([X_Goal], [Y_Proof], Proof),
    concat([X_Gen], [Y_GenProof], GenProof).
```

9.6.3 实验例子

这里通过“自杀”这个例子来说明如何实现 EBG。

输入：

目标概念：suicide(x).

领域理论：一组子句或称规则.

suicide(x) :- kill(x, x).

kill(A, B) :- hate(A, B),

possess(A, C),

weapon(C).

hate(A, A) :- depressed(A).

possess(A, C) :- buy(A, C).

weapon(Z) :- gun(Z).

训练例：一组事实子句。

depressed(john).

buy(john, gun1).

gun(gun1).

suicide(john).

可操作性标准：暂时简单地处理为静态标准。

operational(depressed).

operational(gun).

operational(buy).

系统首先建立关于 suicide(john)的解释结构(见图 9.8)。由 Prolog 程序的解释机理可以得知：建立解释结构采用的是 Top-Down 的推理策略。每次运用规则时，都相应地将规则变量化之后用于一般化的解释结构上，从而在建立 suicide(john)的解释结构的同时，得到 suicide(x)的泛化解释结构。具体过程如图 9.9 所示。

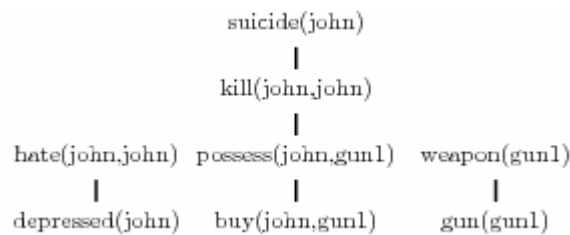


图 9.8 suicide(john)的解释结构

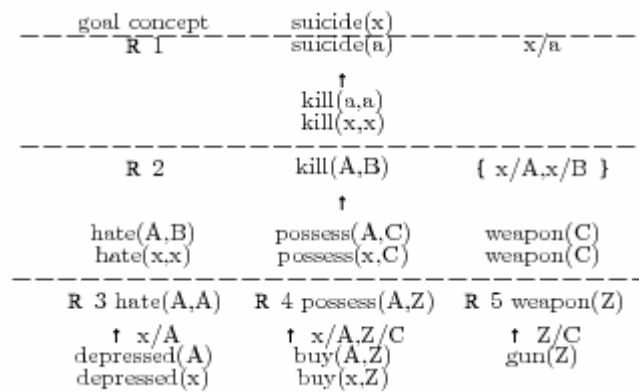


图 9.9 目标概念 suicide(x) 的泛化过程

图 9.9 的泛化过程是通过回归实现的，其中要将谓词完成常量变量化这一过程。

从另一角度考虑实现 EBG，可以两条线推进，一条是建立 suicide(john) 的具体的证明结构，另一条是从 suicide(x) 出发，得到泛化的解释结构。两条线之间的联系在于具体例子选用的规则就是泛化解释中要用的规则。也就是说，建立具体例子解释时需要搜索问题空间，以试图找到合适的可用规则，而建立泛化解释则可免去搜索过程，直接使用训练例解释时使用的规则。

9.7 基于知识块的 SOAR 系统

50 年代末，对神经元的模拟中发明了用一种符号来标记另一些符号的存储结构模型，这是早期的记忆块(chunks)概念。在象棋大师的头脑中就保存着在各种情况下对弈经验的存储块。80 年代初，Newell 和 Rosenbloom 认为，通过获取任务环境中关于模型问题的知识，可以改进系统的性能，记忆块可以作为对人类行为进行模拟的模型基础。通过观察问题求解过程，获取经验记忆块，用其代替各个子目标中的复杂过程，可以明显提高系统求解的速度。由此奠定了经验学习的基础。

1987 年，斯坦福大学 Andrew Golding 和 Paul S. Rosenbloom 与密西根大学的 John E. Laird 开发了 SOAR 系统。它的学习机制是由外部专家的指导来学习一般的搜索控制知识。外部指导可以是直接劝告，也可以是给出一个直观的简单问题。系统把外部指导给定的高水平信息转化为内部表示，并学习搜索记忆块。图 9.10 给出了 SOAR 的框图。

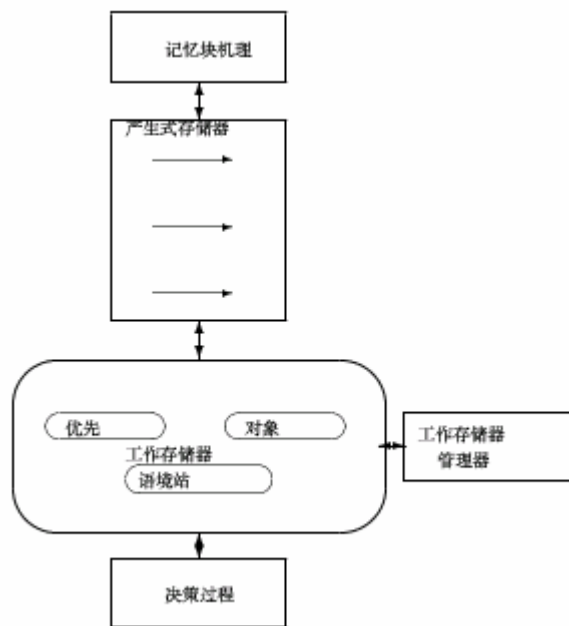


图 9.10 SOAR 的体系结构

产生式存储器和决策过程形成处理结构。产生式存储器中存放产生式规则，它进行搜索控制决策分为两个阶段。第一阶段是详细推敲阶段，所有规则被并行地用于工作存储器，判断优先权，决定哪部分语境进行改变，怎样改变。第二阶段是决策阶段，决定语境栈中要改变的部分和对象。

有时因知识不完善，无法唯一决策而进入困境(impass)。为了解决困境，SOAR 自动建立子目标和新的语境来解决困境。选择问题空间、状态和操作符，创建子目标。对每个困境都产生子目标，SOAR 问题求解能力可以排除困境。为了说明 SOAR 的工作过程，现以九宫问题为例，其初始状态和目标状态如图 9.11 所示。

初始状态			目标状态		
2	3	1	1	2	3
	8	4	8		4
7	6	5	7	6	5

图 9.11 九宫问题的初始状态和目标状态

该问题求解的过程如图 9.12 所示。

1. G_1 solve-eight puzzle
2. P_1 eight-puzzle sd
3. S_1

2	3	1
	8	4
7	6	5

4. O_1 place-blank
5. $\Rightarrow G_2$ (resolve-no-change)
6. P_2 eight-puzzle
7. S_1
8. $\Rightarrow G_3$ (resolve-tie-operator
9. P_3 tie
10. S_2 (left,up,down)
11. O_5 evaluate-object(O_2 (left))
12. $\Rightarrow G_4$ (resolve-no-change)
13. P_2 eight-puzzle
14. S_1
15. Q_2 left
16. S_3

2	3	1
8		4
7	6	5

17. O_2 left
18. S_4
19. S_4
20. O_8 place-1

图 9.12 SOAR 求解九宫问题的过程}

每当问题求解器不能顺利求解时，系统就进入劝告问题空间请求专家指导。专家以两种方式给以指导。一种是直接指令式，这时系统展开所有的算子以及当时的状态。由专家根据情况指定一个算子。指定的算子要经过评估，即由系统建立一个子目标，用专家指定的算子求解。如果有解，则评估确认该算子是可行的，系统便接受该指令，并返回去求证用此算子求解的过程为何是正确的。总结求证过程，从而学到使用专家劝告的一般条件，即记忆块。

另一种是间接的简单直观形式，这时系统先把原问题按语法分解成树结构的内部表示，并附上初始状态。然后请求专家劝告。专家通过外部指令给出一直观的简单音量。它应该与原问题近似，系统建立一个子目标来求解这个简单问题。求解完后就得到算子序列，学习机制通过每个子目标求解过程学到记忆块。用记忆块直接求解原问题，不再需要请求指导。

SOAR 系统中的记忆块学习机制是学习的关键。它使用工作存储单元 (working-memory-elements) 来收集条件并构造记忆块。当系统为评估专家的劝告，或为求解简单问题而建立一个子目标时，首先将当时的状态存入工作存储单元 w-m-e。当子目标得到解以后，系统从 w-m-e 中取出子目标的初始状态，删去与算子或求解简单问题所得出的解算子作为结论动作。由此生成产生式规则，这就是记忆块。如果子目标与原问题的子目标充分类似，记忆块就会被直接应用到原问题上，学习策略就把在一个问题上学到的经验用到另一个问题上。

记忆块形成的过程可以说是依据对于子目标的解释。而请示外部指导，然后将专家指令或直观的简单问题转化为机器可执行的形式，这运用了传授学习的方法。最后，求解直观的简单问题得到的经验(即记忆块)被用到原问题，这涉及到类比学习的某些思想。因此可以说，SOAR 系统中的学习是几种学习方法的综合应用。

9.8 可操作性标准

可操作性标准 (operationality criterion) 是 EBG 的一个输入要求。关于可操作性标准, Mitchell, Keller 和 Kedar-Cabelli 指出:“概念描述应该表示为描述训练例的那些谓词或其他从领域理论中挑选出来的易于估值的谓词。”显然,这种处理很直观、简单,只需在领域知识中增加一个谓词 operational (pid) 说明哪些谓词是可操作的即可。但是,这种处理是静态的,远远不能满足实用学习系统的要求。所以,如何把可操作性标准处理成动态的,已经越来越为研究人员重视。比如可以引入定理证明机制,从而动态地确定可操作性标准,甚至可以用 EBG 技术动态地改变可操作性标准。这样动态地处理,要求引入推理机制,而不再只是列出可操作的谓词而已。既然要推理,就需一些规则、事实(前提),这就涉及到元级 (meta-level) 程序设计的一些技术,要求系统能处理元规则。但 Turbo Prolog 不直接支持元程序设计,如果把可操作性标准做成动态的,首先要在系统中实现一些元级谓词,使系统支持元级知识才行。所以,我们暂时把可操作性标准处理为静态的,用数据库谓词,把事先规定好的一些谓词名,存在内部数据库中。

提出可操作性问题的目的是希望学到的概念描述有利于提高系统运行效率。一种观点认为:如果一个概念描述能有效地用于识别相应概念的例子,则它是可操作的。但是概念描述不只用于识别例子,还有空间指标等。因此,上述观点不完善。

Keller 将可操作性定义如下:

如果给定:

- (1) 一个概念描述。
- (2) 一个执行系统,它利用概念描述改善执行情况。
- (3) 改善执行系统的各种要求,应明确各要求的类型和程度。

那么若满足下列二条件则该概念描述是可操作的:

- (1) 可用性: 执行系统可以使用该概念描述。
- (2) 效用性: 执行系统使用概念描述时,系统的运行得到要求的改善。

例如对 Safe-to-Stack(x, y) 的一个泛化解释结构如图 9.13 所示。

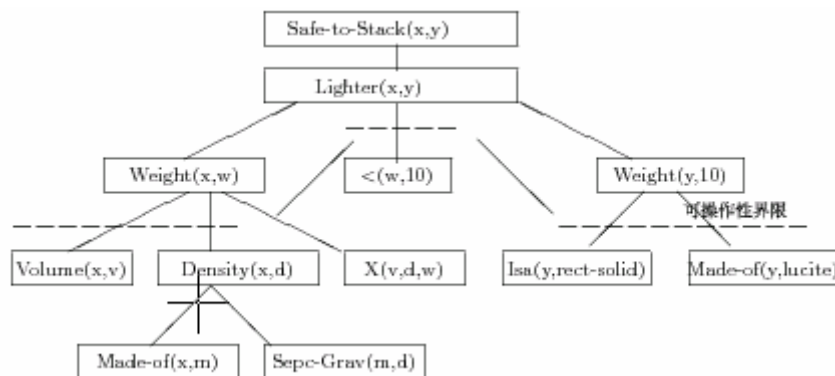


图 9.13 可操作性修剪

解释结构中的叶节点都是可操作的,如 Made-of(x, m) 和 Spec-Grav(m, d)。某些中间节点也可能是可操作的,在图中的 Density(x, d) 是可操作的。于是得到图中的可操作性界限,图中虚线以下各节点是可操作的。对于可操作的中间节点,可以从解释结构中删掉它的子解释。如图中可删去 Made-of(x, m) 和 Spec-Grav(m, d)。这称为“可操作性修剪”。图 9.13 的解释经过修剪后得到下列规则:

$$\text{Volume}(x, v) \wedge \text{Density}(x, d) \wedge X(v, d, w)$$

$$\wedge \text{Isa}(y, \text{rect-solid}) \wedge \text{Made-of}(y, \text{lucite})$$

$$\wedge < (w, 10) \rightarrow \text{Safe-to-Stack}(x, y)$$

关于可操作性标准的处理，是一个较复杂的问题，仅可操作性标准的确定就可能涉及多方面的问题。这个问题处理的好坏又将直接影响学习系统的好坏，所以它在学习系统中是一个相当重要的问题。

多数系统对可操作性标准没有精确的定义。许多系统认为可操作性标准是独立的、静态的。DeJong & Mooney 等人则提出静态标准是不能满足需要的，并且用例子说明可操作性应该是证明结构的函数，也就是说，可操作性标准应该是动态的。比如，可以定义一般化的解释结构的终结结点的谓词是可操作的；也可以应用一个定理证明机制来决定可操作性。进一步讲，甚至可以运用 EBG 来决定可操作性。不管怎样，系统的可操作性标准应该反映计算的代价及学习的得益，尤其应该确保时间代价不能高于得益。但是，最初的基于解释的学习却忽略了这一点，常常认为 EBL 的问题求解系统学到的知识必定能改进系统的性能，但这种认识大概是过于乐观，过于简单化了。一般来说，EBL 实现的是启发式策略，不能保证在所有环境下都能使性能得到提高。例如 STRIPS MACROPS 的学习实际上降低了效率。如果测试 macro-operator 的前提所需的累积时间超过了使用 macro-operator 减少搜索而节省的时间，那么总的性能是降低了。所以，认为可操作性完全独立于系统显然是不符合所有情况的。

Mitchell 等提出的 EBG 中用显式的可操作性标准测试解释的可应用性，这当然是很直观的处理方法。然而，如 Dejong 和 Mooney 所指出的，Mitchell 等提出的可操作性标准只能包括一小部分谓词，而且仅保证新知识能够被直接评价，不保证新知识是否有用。如在 EBG 的例子中，可操作性标准仅仅要求解释必须进行到能直接计算或直接观察到的属性才停止。后来的 META-LEX 在这个问题上考虑了学习的环境，使测试一个属性的代价随系统的知识发生变化。于是，可操作性标准就不再是静态的，而成为动态的了。PRODIGY 更进一步改进了可操作性标准的定义。

9.8.1 PRODIGY 的效用问题

PRODIGY 系统是通过分析经验，与专家交互来获取知识的。该系统学习的是与问题求解有关的目标概念，比如学习 successful, unsuccessful, preferred 这些目标概念。典型的问题求解领域是机车间的调度问题：要求将原材料通过 LATHE, CLAMP, POLISH 等操作，变成产品。PRODIGY 的学习任务是要总结出在什么情况下使用某个操作是 successful 的或 unsuccessful 的。这个系统的一个特点是纳入了一些控制规则，用以提高搜索效率，改善问题解的质量，并且能够使某些原来从效率角度考虑不加搜索的路径得到搜索，PRODIGY 在处理可操作性标准时，要求学到的知识能提高问题求解机制的效率，要求学到的控制知识不仅只是能用，而且应当是有用的，因此，这里的可操作性就需包括可用性：学到的控制规则不但要能够执行，而且要能够确实改进系统的性能。具体一点，控制规则的可用性可以定义如下：

$$\text{Utility} = (\text{AvrSavings} \times \text{ApplicFreq}) - \text{AvrMatchCost} \quad (9.1)$$

其中：

AvrMatchCost = 匹配该规则的平均耗费
 AvrSavings = 应用该规则时，平均节约的时间
 ApplicFreq = 应用该规则的频度

系统学到一条控制规则之后，PRODIGY 就会在随后处理问题的过程中保留使用该规则的统计信息，用于决定其可用性。如果规则的可用性是负值，这条规则就要被放弃，从这个角

度看，在 PRODIGY 系统中可操作性很大程度上是由可用性表示的，由此看来，可操作性能够被量化，可以不预先确定，而是在系统执行过程中动态地确定其值。

将可用性纳入可操作性标准，有助于建立通用的基于解释的学习系统的研究，使得系统在估价可操作性的能力上更加成熟。以前的系统，大多从如何有效地识别训练例的角度来定义可操作性，但是对于通用定义这是远远不够的。一是这种可操作性定义假设概念描述将用于识别训练例，其实，概念描述还有其他用途，比如训练例的泛化。二是多数系统中使用定义来有效识别训练例时，是假设用执行时间作为性能好坏的度量来估价可操作性的。实际上，还有其他方面的效率可以用于估价性能的好坏，如空间效率。常常有这样的情况出现：从时间效率上看是可操作的，从空间效率上看却不可操作。除了效率以外，还有代价、简单度等等标准与性能有关，也可能影响可操作性。

9.8.2 SOAR 系统的可操作性

SOAR 系统则采用了不同的处理可操作性标准的方法。SOAR 系统是由 Laird, Newell 和 Rosebloom 研制的。SOAR 并不是作为一个 EBL 系统研制开发的，它只是想通过一种独立的学习机制 chunking 做成一个通用的认知结构。Chunking 操作就是在处理每个子目标时总结考查过的信息。这种机制与 EBL 极为相似，而且通过 Chunking 似乎可以在 SOAR 中实现 EBL。Chunking 要求的输入是操作符的线性序列或树形序列。系统的任务是把这一串操作符转化成一个宏操作符（或称 Chunk）。SOAR 系统主要是通过 Chunking 来获取知识的，因此就有意忽略了可用性。一是由于这里假设 Chunking 是自动完成的；二是由于 SOAR 系统的性能是由完成一项任务所需做的抉择次数来衡量的。既然 Chunking 可能减少所需的抉择次数，按照性能公式应该很合算。但实际上，每次抉择本身就可能包含复杂的过程，没有考虑这一点，因而抉择次数并不与实际的主机执行时间成正比。

9.8.3 MRS-EBG 的可操作性

Hirsh 在 MRS 逻辑程序设计系统下实现 EBG 时，由 MRS 的元推理机制想到将可操作性标准处理为可证明的。在 MRS 中，证明策略能够用元级的理论来说明。举例来说，元规则可以用来说明这样的规则：选择一条使用了算术谓词的证明路径的可能性更大一些。MRS 本身包含的谓词就可以说明这种属性，因此很容易说明如下的可操作性规则：

Arithmetic_Predicates(pred(arg1, argn))-operational(pred(arg1, arg5))

Hirsh 认为以前的 EBG 算法，都同时产生例子的解释结构和泛化的解释结构。但是，在产生解释结构时没有进行可操作性的推理，而是在随后的泛化那一步，才决定可操作性并修改解释结构。泛化这一步使用可操作性来对泛化了的解释结构进行删减，然后使用得到的结构来形成规则。这样一来，尽管多数泛化的工作是在解释过程中完成的，但解释和泛化仍然没能结合起来。

在 MRS-EBG 中，可操作性是在解释过程中确定的，一旦某一分支由于没有可操作的定义而终结，就立即进行回溯，寻找该分支的另一能够产生一可操作的概念定义的证明。这样，最终一定能得到目标概念的可操作性描述。

9.8.4 META-LEX 的处理方法

在 Metalex 系统中,可操作性的处理考虑了系统的性能因素。基本想法是通过经验来评估可操作性:在系统中使用概念描述,然后看系统的行为是否达到了事先提出的系统目标。Metalex 要学习的,是在向前搜索中问题求解步骤的集合,或者说是这个集合的描述。如果系统的性能没能达到所希望的水平, Metalex 能够估计出该描述差多少就能说是可执行的了,并且能说明这种搜索方向是否合适。这种估价可操作性的办法是动态的,因为它取决于当前状态和当前的性能目标;另外,这种估价产生出效率的量度和有效度。但是这种处理的代价很高,因为每次需要估价可操作性时都必须测试系统。

可操作性对于基于解释的学习系统是至关重要的。但是,目前检测可操作性的方法决定于能否简化系统的性能假设(这些假设很容易在学习过程中被破坏)。尽管在这方面国内外的研究人员一直在寻找有效的处理方法,但多数只是处于理论研究阶段,应用于实际系统中的尚未达到令人满意的程度。因此,这方面的工作仍有待于进一步的研究。

9.9 不完全领域知识下的解释学习

9.9.1 不完全领域知识

解释学习中一个重要的问题是领域知识。领域知识作为解释学习的前提,要求是完整的、正确的。但是这种要求在现实世界的许多问题中很难得到满足。在实际中往往出现领域知识不完整、有错误等情况。领域知识不能解释训练例,那么现有的 EBG 算法就将失效。

领域知识的不完善性可有下列三种情况:

- 不完整 (incomplete): 领域知识中缺少规则、知识,因而无法给出训练例的解释。
- 不正确 (incorrect): 领域知识中有些规则不合理,因而可能导致训练例的错误解释。
- 难处理的 (intractable): 领域知识过于繁杂,使得建立训练例的解释树所需的资源超出现有机器的资源。

为了解决不完善领域知识的问题,我们对逆归结、基于深层知识等方法,作了有益的尝试。

9.9.2 逆归结方法

一阶谓词逻辑中归结原理是机器定理证明的基础,也是目前解释学习中建立解释的主要途径。

归结原理:设 C_1, C_2 是两个子句,无公用变量, L_1, L_2 分别是 C_1, C_2 中的两个文字,存在一个最一般的合一者 σ , 则子句

$$C = (C_1 - \{L_1\})\sigma \cup (C_2 - \{L_2\})\sigma \quad (9.2)$$

称为 C_1, C_2 的归结子句。

在逆归结 (inverting resolution) 中,讨论已知 C 和 C_1 , 如何得到 C_2 的问题。在命题逻辑中, $\sigma = \Phi$, 所以归结子句的式 (9.2) 可以转换为:

$$C = (C_1 \cup C_2) - \{L_1, L_2\} \quad (9.3)$$

由式 (9.3) 可以得到:

(1) 若 $C_1 \cap C_2 = \emptyset$, 则 $C_2 = (C - C_1) \cup \{L_2\}$

(2) 若 $C_1 \cap C_2 \neq \emptyset$, C_2 需包括 $C_1 - \{L_1\}$ 的任意子集, 因此, 一般情况下有:

$$C_{2i} = (C - C_1) \cup \{L_2\} \cup S_{1i} \quad (9.4)$$

其中 $S_{1i} \in P(C_1 - \{L_1\})$, $P(x)$ 表示集合 x 的幂集。

显然 如果 C_1 中有 n 个文字, C_2 就有 $2n-1$ 个解。

在一阶谓词逻辑中, 令 $\sigma = \theta_1\theta_2$, 其中 θ_1, θ_2 满足下列条件:

(1) θ_1, θ_2 的变量域分别为 C_1 的变量域和 C_2 的变量域,

$$(2) \overline{L_1} \theta_1 = \overline{L_2} \theta_2$$

则可以得到:

$$\begin{aligned} C_2 &= (C - (C_1 - \{L_1\}) \theta_1) \theta_2^{-1} \cup \{\overline{L_1}\} \theta_1 \theta_2^{-1} \\ &= ((C - (C_1 - \{\theta_1\}) \theta_1) \cup \{\overline{L_1}\} \theta_1) \theta_2^{-1} \end{aligned} \quad (9.5)$$

当 C_1 是单元子句, 即 $C_1 = \{L_1\}$ 时, 可以得到

$$C_2 = (C \cup \{\overline{L_1}\} \theta_1) \theta_2^{-1} \quad (9.6)$$

这里 θ_2^{-1} 是逆置换。所谓逆置换, 是指给定项或文字 t 及置换 θ , 存在唯一的逆置换 θ^{-1} , 满足 $t\theta\theta^{-1} = t$ 。进一步讲, 若 $\theta = \{v_1/t_1, \dots, v_n/t_n\}$, 那么,

$$\theta^{-1} = \{(t_1, \{P_{1,1}, \dots, P_{1,m_1}\})/v_1, \dots, (t_n, \{P_{n,1}, \dots, P_{n,m_n}\})/v_n\} \quad (9.7)$$

其中 P_{i,m_j} 是 t 中变量 v_i 的位置。逆归结就是将 t 中 $\{P_{i,1}, \dots, P_{i,m_i}\}$ 处的所有 t_i 都换为 v_i 。

解释学习是从目标概念的初始描述出发, 利用领域知识 (以知识库形式存储), 建立关于训练实例的解释树。这个过程通常用目标驱动的推理方式。当解释过程由于缺乏某条规则而无法继续下去时, 学习过程即告失败。这里采用逆归结方法克服这个问题 [马海波 1990]。

我们采用产生式规则表示领域知识 (知识库), 算法需要对知识库进行预处理, 产生一个规则间的依赖表。这样, 对算法而言, 拥有的领域知识包括一组产生式规则和一个依赖表。

依赖表是知识库中规则、谓词间联系的一种表示形式, 它包含着规则, 谓词间相关的语义信息。我们通过一个简单的例子来说明依赖表的结构。

知识库:

Rule 1. Sentence (S0, S) :- noun-phrase(S0, S1),
verb-phrase(S1, S).

Rule 2. noun-phrase (S0, S) :- determiner(S0, S1),
noun(S1, S).

Rule 3. noun-phrase (S0, S) :- name(S0, S).

Rule 4. verb-phrase (S0, S) :- intransitive-verb(S0, S).

依赖表如下:

谓词符号	Head	Body	基本谓词
sentence	1		intransitive-verb, name, determiner, noun.
noun-phrase	2,3	1	name, determiner, noun.
verb-phrase	4	1	intransitive-verb.

其中基本谓词就是满足可操作性标准的谓词，第一列则全是非可操作的谓词。至于构造此依赖表的算法，这里不作详细叙述。下面给出两个算法，完成解释学习过程。

算法 9.3 解释树算法生成算法。

- (1) 对目标概念开始做逆向推理，逐渐扩展解释树，该解释树是一棵与或树。
- (2) 遇到失败结点，调用逆归结算法。
- (3) 分析得到的完整解释树，进行泛化，得到目标概念的新描述。

算法 9.4 逆归结算法。

- (1) 若当前失败结点 F 是可操作的，回溯；若当前失败结点 F 不可操作，转到(2)。
- (2) 与点 F 的父结点 P 对应的谓词符号为 Pred，利用依赖表，检查 Pred 是否有其他路径（即是否有或结点存在）。没有，转到(4)。
- (3) 对依赖表中 Pred 的其他路径（或结点），检查其对应的可操作谓词（基本谓词）是否被训练实例满足。
 - 若存在满足训练例的路径，则选择此路径，完成解释树，结束算法。
 - 依赖表中 Pred 的其他路径（或结点）对应的可操作谓词（基本属性）与训练例冲突或不为训练例具备，转到(4)。
- (4) 先对结点 P 除结点 F 以外的子结点进行推理，所有子结点处理完之后，回到结点 F。
- (5) 用未使用的训练属性，采用逆归结的方法，得到节点 F 与剩余属性间的一条伪规则，完成整个解释树的建立过程，结束算法。

这里描述的算法，能够在领域知识不完全的情况下进行解释学习，但领域知识的不完全必须是个别规则的缺少，如果领域知识过于零散，未能构成较为完整的知识结构，此算法将无能为力，而需要求助于建立知识库的方法和工具，或者采用诸如归纳、类比等其他学习方法。

最后要指出的一点是，算法虽然完成解释学习，但毕竟由于有规则缺少，处理时引入了不精确因素，因而会不同程度地影响整个解释树的真实性，不能保证解释过程的永真。如果能对解释树加入可信度，在学习的后一步进行概括时考虑到可信度的取值、传递，就可以区别那些完全领域的学习。

9.9.3 基于深层知识方法

故障诊断领域的知识经常是不完善的。如何完善故障诊断的知识库是一个急待解决的问题。因此，我们提出了一种用于精馏系统故障诊断系统中的基于深层知识的学习模型(吕翠英1994)。该模型分四阶段：实例解释、猜想生成、猜想确立和推广。应用此模型可以发现现有知识库的知识不能诊断的故障，在获得新知识以后可进行正确的诊断。使用基于深层知识的学习模型的系统学习过程可概述如下：

- (1) 由环境提供一实例。
- (2) 实例解释阶段：系统使用基于解释的学习方法对该实例进行解释。解释过程有可能成功，有可能失败。成功是指用系统现有的领域知识就可以说明。失败是指用系统现有的领

域知识，无论如何，也不可以说明。成功则进入(5)推广。失败则进入(3)猜想生成。解释用解释树来表示。成功则表示有一棵完整的解释树。失败则表示解释树不完整，树上有断口。

(3) 猜想生成阶段：系统试图确立自身相对于当前实例的知识断口，即确定出在什么地方缺少知识。当找到知识断口后，建立一个有可能弥补该断口的猜想，该猜想的目标端由断口的目标端充任，该猜想的数据端由输入实例的数据端充任。然后进入(4)猜想确立。

(4) 猜想确立阶段：调用深层知识库。在深层知识库中寻找猜想的目标端与数据端之间的某种内部联系，即试图确立猜想的目标端与数据端在深层知识中是否具有共同的属性。确立可能成功，也可能失败。若失败，退回(3)猜想生成，重新获得新的猜想。若成功，进入(5)推广。

(5) 推广：将已确立的猜想加以推广。推广的结果是：得到若干具有广泛适应面的猜想。推广过程分为两个阶段。第一阶段将猜想中的常数最大限度地变量化。第二阶段对猜想执行概念推广，得到一个或多个不同一般级别的概念。

(6) 将推广的结果交执行单元使用。

习 题

1. 何谓基于解释的学习？与归纳学习相比，解释学习的优点和缺点是什么？
2. 简述解释学习涉及的三个不同的空间，以及三个空间的相互关系。
3. 阐述基于解释的泛化过程，并重点说明目标概念、训练例子、领域理论和可操作准则在这种学习过程中的作用。
4. 比较解释泛化学习方法和解释特化学习方法的基本思想。
5. 在你选择的一些问题领域建立基于解释的学习的领域理论。对几个训练实例运用这些理论，说明基于解释的学习程序的行为。
6. 试述可操作性标准的含义，以及将可操作性标准处理成动态的方法。
7. 简述在不完全领域知识下的解释学习解决策略。
8. 假设 $C = \text{Son}(\text{Bob}, \text{Ram})$, $C1 = \neg \text{Daughter}(\text{Bob}, \text{Ram})$ ，使用归纳逻辑程序设计原理，确定子句 $C2$ 。

第十章 强化学习

10.1 概 述

人类通常从与外界环境的交互中学习。所谓强化(reinforcement)学习是指从环境状态到行为映射的学习,以使系统行为从环境中获得的累积奖励值最大。在强化学习中,我们设计算法来把外界环境转化为最大化奖励量的方式的动作。我们并没有直接告诉主体要做什么或者要采取哪个动作,而是主体通过看哪个动作得到了最多的奖励来自己发现。主体的动作的影响不只是立即得到的奖励,而且还影响接下来的动作和最终的奖励。试错搜索(trial-and-error search)和延期强化(delayed reinforcement)这两个特性是强化学习中两个最重要的特性。

强化学习不是通过特殊的学习方法来定义的,而是通过在环境中和响应外界环境的动作来定义的。任何解决这种交互的学习方法都是一个可接受的强化学习方法。强化学习也不是监督学习,在我们有关机器学习的部分我们都可以看出来。在监督学习中,“教师”用实例来直接指导或者训练学习程序。在强化学习中,学习主体自身通过训练,误差和反馈,学习在环境中完成目标的最佳策略。

强化学习技术是从控制理论、统计学、心理学等相关学科发展而来,最早可以追溯到巴甫洛夫的条件反射实验。但直到上世纪八十年代末、九十年代初强化学习技术才在人工智能、机器学习和自动控制等领域中得到广泛研究和应用,并被认为是设计智能系统的核心技术之一。特别是随着强化学习的数学基础研究取得突破性进展后,对强化学习的研究和应用日益开展起来,成为目前机器学习领域的研究热点之一。

强化思想最先来源于心理学的研究。1911年Thorndike提出了效果律(Law of Effect):一定情景下让动物感到舒服的行为,就会与此情景增强联系(强化),当此情景再现时,动物的这种行为也更易再现;相反,让动物感觉不舒服的行为,会减弱与情景的联系,此情景再现时,此行为将很难再现。换个说法,哪种行为会“记住”,会与刺激建立联系,取决于行为产生的效果。

这也称为动物的试错学习(trial-and-error learning),包含两个含义:选择(selectional)和联系(associative),对应计算上的搜索和记忆。所以,1954年,Minsky在他的博士论文中实现了计算上的试错学习。同年,Farley和Clark也在计算上对它进行了研究。强化学习一词最早出现于科技文献是1961年Minsky的论文“Steps Toward Artificial Intelligence”,此后开始广泛使用。1969年,Minsky因在人工智能方面的贡献而获得计算机图灵奖。

Farley和Clark在1994年和1995年用试错学习来做泛化和模式识别,从此开始了强化学习和指导学习的混淆。直到最近,神经网络的书还说它是一个试错学习系统,因为它是通过输出误差来调整权重,但是他们忽略了试错的一个重要特性:选择。

20世纪60、70年代,指导学习得到了广泛的研究,如神经网络、统计学习等。强化学习处于低潮期,一是混淆了强化学习和指导学习,以为在研究前者,实际在研究后者;二是后者相对简单一些,后者有教师——训练例子,而前者没有。

试错学习的研究不象时间差分 and 最优控制学习那样突出。时序差分 (temporal difference) 学习基于时间序列上对同一个量相继两个估计的差, 它的起源是心理学上的次强化物 (secondary reinforcer)。原强化物 (primary reinforcer): 直接满足个体需求的刺激, 如食物等; 次强化物: 经学习而间接使个体满足的刺激物, 如奖状、金钱等。

最早把这个心理学原理引入人工学习系统的还是 1954 年的 Minsky。1959 年, Samuel 在他著名的跳棋游戏中也应用了时序差分的思想。1972 年, Klopff 把试错学习和时序差分结合在一起。1978 年开始, Sutton、Barto、Moore, 包括 Klopff 等对这两者结合开始进行深入研究。

最优控制于 50 年代提出: 为动态系统设计一个控制器, 在从初态转移到终态时, 保证系统的某个性能指标保持最小值 (或最大值)。1953 到 1957 年, Bellman 提出了求解最优控制问题的一个有效方法: 动态规划 (dynamic programming), (另一个有效方法是苏联庞特里雅金等人于 1956-1958 提出的最大值原理)。动态规划在随后的四十年里得到的深入的研究, 特别是在自动控制领域。

Bellman 于 1957 年还提出了最优控制问题的随机离散版本, 就是著名的马尔可夫决策过程 (MDP, Markov decision processe), 1960 年 Howard 提出马尔可夫决策过程的策略迭代方法, 这些都成为现代强化学习的理论基础。

真正把时序差分和最优控制结合在一起的是 1989 年 Watkins 提出了 Q-学习 [Watkins 1989], 也把强化学习的三条主线扭在了一起。1992 年, Tesauro 用强化学习成功了应用到西洋双陆棋 (backgammon) 中, 称为 TD-Gammon [Tesauro 1992]。从此开始了强化学习的深入研究。

10.2 强化学习模型

强化学习的模型如图 10.1 所示, 通过主体与环境的交互进行学习。主体与环境的交互接口包括行动 (Action)、奖励 (Reward) 和状态 (State)。交互过程可以表述为如下形式: 每一步, 主体根据策略选择一个行动执行, 然后感知下一步的状态和即时奖励, 通过经验再修改自己的策略。主体的目标就是最大化长期奖励。

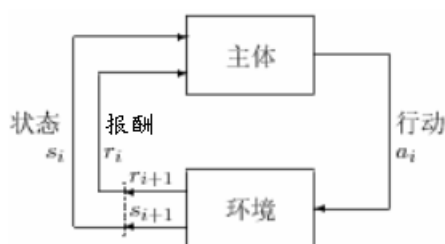


图 10.1 强化学习模型

强化学习系统接受环境状态的输入 s , 根据内部的推理机制, 系统输出相应的行为动作 a 。环境在系统动作作用 a 下, 变迁到新的状态 s' 。系统接受环境新状态的输入, 同时得到环境对于系统的瞬时奖惩反馈 r 。对于强化学习系统来讲, 其目标是学习一个行为策略 $\pi: S \rightarrow A$, 使系统选择的动作能够获得环境奖励的累计值最大。换言之, 系统要最大化 10.1 式, 其中 γ 为折扣因子。在学习过程中, 强化学习技术的基本原理是: 如果系统某个动作导致环境正的奖励, 那么系统以后产生这个动作的趋势便会加强。反之系统产生这个动作的趋势便减弱。这和生理学中的条件反射原理是接近的。

$$\sum_{i=0}^{\infty} \gamma^i r_{t+i} \quad 0 < \gamma \leq 1 \quad (10.1)$$

如果假定环境是马尔可夫型的，则顺序型强化学习问题可以通过马尔可夫决策过程建模。下面首先给出马尔可夫决策过程的形式化定义。

马尔可夫决策过程 由四元组 $\langle S, A, R, P \rangle$ 定义。包含一个环境状态集 S ，系统行为集合 A ，奖励函数 $R: S \times A \rightarrow \mathcal{R}$ 和状态转移函数 $P: S \times A \rightarrow PD(S)$ 。记 $R(s, a, s')$ 为系统在状态 s 采用 a 动作使环境状态转移到 s' 获得的瞬时奖励值；记 $P(s, a, s')$ 为系统在状态 s 采用 a 动作使环境状态转移到 s' 的概率。

马尔可夫决策过程的本质是：当前状态向下一状态转移的概率和奖励值只取决于当前状态和选择的动作，而与历史状态和历史动作无关。因此在已知状态转移概率函数 P 和奖励函数 R 的环境模型知识下，可以采用动态规划技术求解最优策略。而强化学习着重研究在 P 函数和 R 函数未知的情况下，系统如何学习最优行为策略。

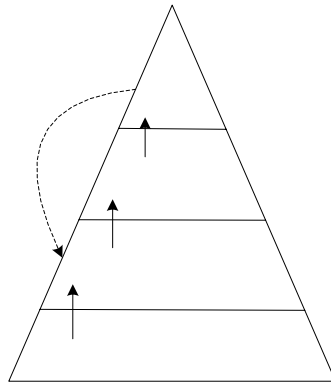


图 10.2 强化学习四要素

为解决这个问题，图 10.2 中给出强化学习四个关键要素之间的关系，即策略 π ，状态值映射 V ，奖励函数 r ，和一个环境的模型(通常情况)。四要素关系自底向上呈金字塔结构。策略定义在任何给定时刻学习主体的选择和动作的方法。这样，策略可以通过一组产生式规则或者一个简单的查找表来表示。像刚才指出的，特定情况下的策略可能也是广泛搜索，查询一个模型或计划过程的结果。它也可以是随机的。策略是学习主体中重要的组成部分，因为它自身在任何时刻足以产生动作。

奖励函数 R_t 定义了时刻 t 问题的状态/目标关系。它把每个动作，或更精细的每个状态-响应对，映射为一个奖励量，以指出那个状态完成目标的愿望的大小。强化学习中的主体有最大化总的奖励的任务，这个奖励是它在完成任务时所得到的。

赋值函数 V 是环境中每个状态的一个属性，它指出对从这个状态继续下去的动作系统可以期望的奖励。奖励函数度量状态-响应对的立即的期望值，而赋值函数指出环境中一个状态的长期的期望值。一个状态从它自己内在的品质和可能紧接着它的状态的品质来得到值，策略就是在这些状态下的奖励。例如，一个状态/动作可能有一个低的立即的奖励，但有一个较高的值，因为通常紧跟它的状态产生一个较高的奖励。一个低的价值可能同样意味着状态不与成功的解路径相联系。

如果没有奖励函数，就没有值，估计值的惟一目的是为了获取更多的奖励。但是，状态值函数决定时，是值最使我们感兴趣，因为值指出带来最高的回报的状态和状态的组合。但是，确定值比确定奖励困难。奖励由环境直接给定，而值是估计得到的，然后随着时间推移根据成功和失败重新估计值。事实上，强化学习中最重要也是最难的方面是创建一个有效的确定值

瞬时奖惩

的方法。

强化学习的环境模型是抓住环境行为的方面的一个机制。模型让我们在没有实际试验它们的情况下估计未来可能的动作。基于模型的计划是强化学习案例的一个新的补充，因为早期的系统趋向于基于纯粹的一个主体的试验和误差来产生奖励和值参数。

系统所面临的环境由环境模型定义，但由于模型中 P 函数和 R 函数未知，系统只能够依赖于每次试错所获得的瞬时奖励来选择策略。但由于在选择行为策略过程中，要考虑到环境模型的不确定性和目标的长远性，因此在策略和瞬时奖励之间构造值函数（即状态的效用函数），用于策略的选择。

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = r_{t+1} + \gamma R_{t+1} \quad (10.2)$$

$$V^\pi(s) = E_\pi \{R_t | s_t = s\} = E_\pi \{r_{t+1} + \gamma V(s_{t+1}) | s_t = s\} = \sum_a \pi(s, a) \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V^\pi(s')] \quad (10.3)$$

首先通过（10.2）式构造一个返回函数 R_t ，用于反映系统在某个策略 π 指导下的一次学习循环中，从 s_t 状态往后所获得的所有奖励的累计折扣和。由于环境是不确定的，系统在某个策略 π 指导下的每一次学习循环中所得到的 R_t 有可能是不同的。因此在 s 状态下的值函数要考虑不同学习循环中所有返回函数的数学期望。因此在 π 策略下，系统在 s 状态下的值函数由（10.3）式定义，其反映了如果系统遵循 π 策略，所能获得的期望的累计奖励折扣和。

根据 Bellman 最优策略公式，在最优策略 π^* 下，系统在 s 状态下的值函数由（10.4）式定义。

$$V^*(s) = \max_{a \in A(s)} E \{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\} = \max_{a \in A(s)} \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V^*(s')] \quad (10.4)$$

在动态规划技术中，在已知状态转移概率函数 P 和奖励函数 R 的环境模型知识前提下，从任意设定的策略 π_0 出发，可以采用策略迭代的方法（式 10.5 和式 10.6）逼近最优的 V^* 和 π^* 。式 10.5 和式 10.6 中的 k 为迭代步数。

$$\pi_k(s) = \arg \max_a \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V^{\pi_{k-1}}(s')] \quad (10.5)$$

$$V^{\pi_k}(s) \leftarrow \sum_a \pi_{k-1}(s, a) \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V^{\pi_{k-1}}(s')] \quad (10.6)$$

但由于强化学习中， P 函数和 R 函数未知，系统无法直接通过（10.5）式、（10.6）式进行值函数计算。因而实际中常采用逼近的方法进行值函数的估计，其中最主要的方法之一是蒙特卡罗(Monte Carlo)采样，如（10.7）式。其中 R_t 是指当系统采用某种策略 π ，从 s_t 状态出发获得的真实的累计折扣奖励值。保持 π 策略不变，在每次学习循环中重复地使用（10.7）式，下式将逼近（10.3）式。

$$V(s_t) \leftarrow V(s_t) + \alpha [R_t - V(s_t)] \quad (10.7)$$

结合蒙特卡罗方法和动态规划技术，式(10.8)给出强化学习中时间差分学习(TD, Temporal difference)的值函数迭代公式。

$$V(s_t) \leftarrow V(s_t) + \alpha [r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \quad (10.8)$$

10.3 动态规划

动态规划(dynamic programming)的方法通过从后继状态回溯到前驱状态来计算赋值函数。动态规划的方法基于下一个状态分布的模型来接连的更新状态。强化学习的动态规划的方法是基于这样一个事实：对任何策略 π 和任何状态 s ，有(10.9)式迭代的一致等式成立：

$$V^\pi(s) = \sum_a \pi(a|s) \times \sum_{s'} \pi(s \rightarrow s' | a) \times (R^a(s \rightarrow s') + \gamma V^\pi(s')) \quad (10.9)$$

$\pi(a|s)$ 是给定在随机策略 π 下状态 s 时动作 a 的概率。 $\pi(s \rightarrow s' | a)$ 是在动作 a 下状态 s 转到状态 s' 的概率。这就是对 V^π 的 Bellman(1957) 等式。它表示了一个状态的值和它的后继状态迭代计算的值之间的关系。在图 10.3 中，我们给出第一步计算，从状态 s 我们向前看三个可能的后继。对策略 π ，动作 a 出现的概率为 $\pi(a|s)$ 。从这三个状态中的每个状态，环境可能响应其中的一个状态，我们说 s' 有奖励 r 。Bellman 等式对这些概率取平均值，不采用每个出现的可能性进行加权。它指出起始状态的值必须期待的下一个状态的折扣值， γ ，加上从这一路径产生的奖励。在动态规划中，如果 n 和 m 表示状态和动作的数目，虽然确定性策略的总数目为 nm ，一个动态规划方法能保证在多项式时间找到最优策略。在这个意义上，动态规划比任何策略空间中的直接搜索指数级的快，关于状态数和行动数的多项式时间。但如果状态数是根据某些变量指数增长的，当然也会出现维度灾难了

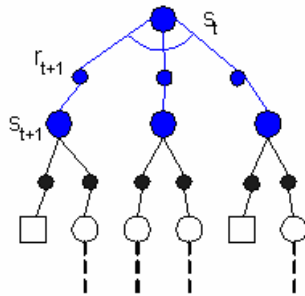


图 10.3 动态规划方法

在动态规划中，由 (10.4) 式可以得到 (10.10) 式的迭代公式：

$$V_{t+1}(s) \leftarrow \text{MAX}_a \sum_{s'} P_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V_t(s')] \quad (10.10)$$

当 $t \rightarrow \infty$ 时， $V_t(s) \rightarrow V^*(s)$ ，重复对每个状态 s 进行迭代，直到 $|\Delta V|$ 小于一个小正数，得到对每个状态长远奖励的评价。

典型的动态规划模型作用有限，很多问题很难给出环境的完整模型。仿真机器人足球就是这样的问题，可以采用实时动态规划方法解决这个问题。在实时动态规划中不需要事先给出环境模型，而是在真实的环境中不断测试，得到环境模型。可以采用反传神经网络实现对状态泛化，网络的输入单元是环境的状态 s ，网络的输出是对该状态的评价 $V(s)$ 。

10.4 蒙特卡罗方法

蒙特卡罗方法不需要一个完整的模型。而是它们对状态的整个轨道进行抽样，基于抽样

点的最终结果来更新赋值函数。蒙特卡罗方法不需要经验，即从与环境联机的或者模拟的交互中抽样状态、动作和奖励的序列。联机的经验是令人感兴趣的，因为它不需要环境的先验知识，却仍然可以是最优的。从模拟的经验中学习功能也很强大。它需要一个模型，但它可以是生成的而不是分析的，即一个模型可以生成轨道却不能计算明确的概率。于是，它不需要产生在动态规划中要求的所有可能转变的完整的概率分布。

于是，蒙特卡罗方法通过对抽样返回值取平均的方法来解决强化学习问题。为了确保良好定义的返回值，蒙特卡罗方法定义为完全抽样，即所有的抽样点必须最终终止。而且，只有当一个抽样点结束，估计值和策略才会改变。这样，在抽样点意义上说，蒙特卡罗方法在一个抽样点上是增加的，而不是逐步的。图 10.4 中，蒙特卡罗方法采样一次学习循环所获得的奖惩返回值。然后通过多次学习，用实际获得的奖惩返回值去逼近真实的状态值函数。

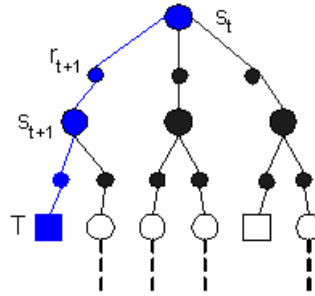


图 10.4 蒙特卡罗方法

蒙特卡罗方法没有环境模型，根据经验学习。只考虑最终任务，任务结束后对所有的回报进行平均。给定策略 π ，计算 V^π 。方法：根据策略 π 产生从开始到任务完成的一个状态序列；对序列中第一次（或每次）出现的状态 s_t ，获得它的长期奖励 $R_t(s_t)$ ；把 $R_t(s_t)$ 加入列表 R_{s_t} ， $V(s_t) \leftarrow \text{average}(R_{s_t})$ 。

列表可以改成增量实现（Incremental Implementation），

$$\begin{aligned} V(s_t) &\leftarrow V(s_t) + \frac{R_t(s_t) - V(s_t)}{N_{s_t} + 1} \\ N_{s_t} &\leftarrow N_{s_t} + 1 \end{aligned} \quad (10.11)$$

策略在线蒙特卡罗控制时，策略评估和改进时使用相同的随机策略，如

$$\begin{aligned} a^* &\leftarrow \arg \max_a Q(s, a) \\ \pi(s, a) &\leftarrow \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A(s)|} & a = a^* \\ \frac{\epsilon}{|A(s)|} & a \neq a^* \end{cases} \end{aligned} \quad (10.12)$$

假定主体与环境交互学习过程中，发现某些动作的效果较好，那么主体在下一决策中该选择什么样的动作呢？一种考虑是充分利用现有的知识，选择当前认为最好的动作，但这样有一个缺点：也许还有更好的动作没有发现；反之，如果主体每次都测试新的动作，将导致有学习没进步，显然不是我们希望的。在利用现有知识和探索新行动之间应做出适当的权衡，称为权衡探索和利用。主要有两种方法，一个是前面用到的 ϵ -贪心法；另一种是模拟退火，每个动作的选择概率和主体对它的评价有关：

$$p(a|s) = \frac{e^{Q(s,a)/T}}{\sum_{a'} e^{Q(s,a')/T}} \quad (10.13)$$

10.5 时序差分学习

时序差分学习中没有环境模型，根据经验学习。每步进行迭代，不需要等任务完成。预测模型的控制算法，根据历史信息判断将来的输入和输出，强调模型的函数而非模型的结构。时序差分方法和蒙特卡罗方法类似，仍然采样一次学习循环中获得的瞬时奖惩反馈，但同时类似与动态规划方法采用自举方法估计状态的值函数。然后通过多次迭代学习，去逼近真实的状态值函数。

时序差分学习技术中经典的 TD(0)学习算法如下。

算法 10.1 TD(0)学习算法

Initialize $V(s)$ arbitrarily, π to the policy to be evaluated

Repeat (for each episode)

Initialize s

Repeat (for each step of episode)

Choose a from s using policy π derived from V (e.g., ϵ -greedy)

Take action a , observe r, s'

$$V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$$

$$s \leftarrow s'$$

Until s is terminal

TD(0)学习算法事实上包含了两个步骤，其一是从当前学习循环的值函数确定新的行为策略；其二是在新的行为策略指导下，通过所获得的瞬时奖惩值对该策略进行评估。学习循环过程为：

$$v_0 \rightarrow \pi_1 \rightarrow v_1 \rightarrow \pi_2 \rightarrow \cdots \rightarrow v^* \rightarrow \pi^* \rightarrow v^*$$

直到值函数和策略收敛。在时序差分学习中，计算值函数方法如图 10.5 所示。

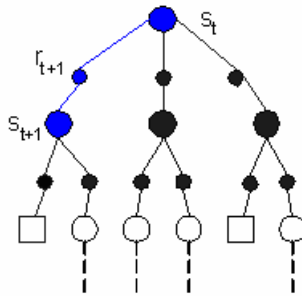


图 10.5 时序差分计算值函数方法

用九宫游戏来说明强化学习算法。首先，必须建立一个数的表，每个数表示一个游戏可能的状态。这些状态值的数反映从这个状态获胜可能性的当前估计。这会支持一个必须赢的策略，也就是说，或者对手赢，或者平局算我们输。这种平算输的方法允许我们建立聚焦于赢的策略。要抓住一个实际选手的技巧，而不是一些理想化选手的完美信息。这样，我们初始化表，用 1 表示我们能赢的每个位置，用 0 表示平或输的位置，用 0.5 表示其余位置，它反映了我们初始时估计从这些状态有 50%的可能性赢。

我们现在与这个对手玩这个游戏。为简单起见，假定我们是◇，我们的对手是 O。图 10.6

反映了游戏中一种可能的移动序列，既有考虑到的又有选择的移动。为了产生一步移动，首先考虑从当前状态一步合法移动到的每个状态，即任何◇可能移动的开放状态。我们查找表中保存的那个状态的当前值。在大多数时刻，我们可以做一步贪婪的移动，即取有最佳赋值函数的状态。偶尔，我们会做一步探测的移动，从其他状态中随机选取一个状态。这些探测的移动是为了考虑在游戏情形中可能看不到的一些选择，来扩大可能值的最优化。

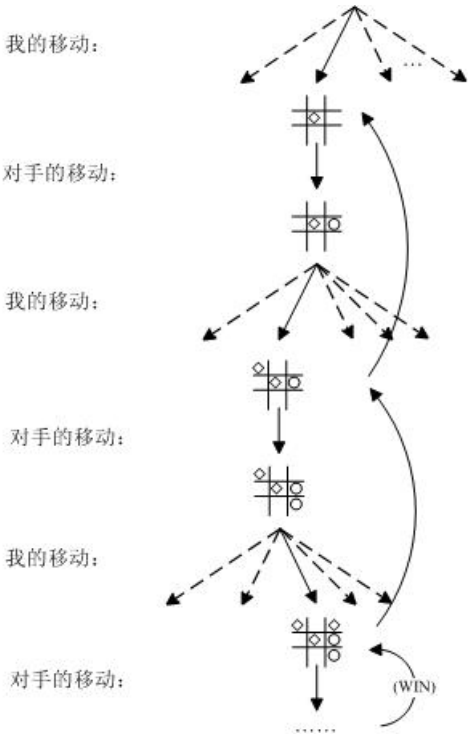


图 10.6 九宫游戏的一个移动序列。向下指向棋盘的虚箭头表示可能的移动选择，向上的实箭头表示当奖励函数改变状态值时的奖励

在我们玩游戏时，我们改变选择的每个状态的赋值函数。我们试图使它们的最新值反映它们在成功路径上的可能性。在前面我们把这个称为一个状态的奖励函数。为了做到这一点，我们退回到我们已经选择的一个状态的值，把它作为我们下一个要选择的的状态的值的函数。如图 10.6 中“向上指”的箭头所示，这个退回的动作越过我们对手的选择，但它确实反映了指导我们选择下一个状态的值的集合。这样，我们选择的前一个状态的当前值就被修正，给予奖励来更好反映后来状态的值（并且最终，当然是反映赢或输的值）。我们通常是通过移动前一个状态一部分差分值来完成这个任务，这个差分值是前一个状态自身与我们选择的新状态之间的差异值。这个部分度量，称为步长参数，它通过等式中的乘数 c 来反映：

$$V(s_n) = S(s_n) + c(V(s_{n+1}) - V(s_n)) \quad (10.14)$$

在这个等式中， s_n 表示在时刻 n 选择的的状态， s_{n+1} 表示在时刻 $n+1$ 选择的的状态。这个更新的等式是时序差分学习规则的一个例子，因为改变是在两个不同时刻 n 和 $n+1$ 估计值的差分 $V(s_{n+1}) - V(s_n)$ 的一个方程。在下一节，进一步讨论这些学习规则。

时序差分方程对九宫游戏执行得非常好。我们想随着时间来减小步长参数，目的是为了伴随着系统的学习，对状态值进行连续减小的调整。给定我们的对手，这能保证每个状态的赋值函数收敛于赢的概率。还有，除了周期性的探索式的移动，做出的选择实际上是最佳的移动，即对这个对手最佳的策略。但是，更令人感兴趣的是这样一个事实：如果步长从未真正减到 0，则这个策略会持续改变来反映对手玩时的任何改变/改进。

我们的九宫游戏解释了强化学习的很多重要特征。首先，有在与环境交互时的学习，这

里的环境是我们的对手。第二，有(反映在很多目标状态上的)清晰的目标和最佳行为，这需要计划和预做准备，以便为特定移动的延期效应保留余地。例如，强化学习算法有效地建立对低级对手的多步移动的计策。这是强化学习的一个重要特征，在没有对手的清晰模型或不进行扩展搜索的情况下，预做准备和计划的效果可以在实际中完成。

在我们的九宫游戏的例子中，学习初始时除了游戏规则没有其他先验知识（我们只是把所有的非终态的状态初始化为 0.5）。强化学习不需要这种“新的开始”的观点。任何能用的先验知识可以使它成为初始状态值的组成部分。处理没有可用信息的状态也是可能的。最后，如果一个情况的模型是可用的，则结果模型所依据的信息可以用于状态的值。但是，重要的是要记住强化学习可以用于以下任一种情况：不需要模型，但如果模型能用或者模型可以被学习到，则可以使用模型。

在九宫游戏的例子中，奖励是随着每个状态-动作的决定分期付给的。我们的主体是近视的，它只考虑最大化立即的奖励。实际上，如果我们使用强化学习进行更深入的预先准备，我们将需要度量最终奖励的折扣回报(discounted return)。我们令折扣率 γ 表示一个未来奖励的当前值：未来 k 个时刻步得到的奖励的价值是立即得到的奖励的价值的 γ^{k-1} 倍。

九宫游戏是两人游戏的一个例子。强化学习还可以用于没有对手，而只是从环境得到反馈的情形。九宫游戏的例子的状态空间还是有限的（实际上相当小）。强化学习还可以用于当状态空间很大，或者甚至是无限的时候。

10.6 Q 学习

在 Q 学习中，Q 是状态-动作对到学习到的值的一个函数。对所有的状态和动作：

$Q: (\text{state} \times \text{action}) \rightarrow \text{value}$

对 Q 学习中的一步：

$$Q(s_t, a_t) \leftarrow (1 - c) \times Q(s_t, a_t) + c \times [r_{t+1} + \gamma \text{MAX}_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (10.15)$$

其中 c 和 γ 都 ≤ 1 ， r_{t+1} 是状态 s_{t+1} 的奖励。我们可以在图 10.7b 中看到 Q 学习的方法，它与图 10.7a 不同，它的开始结点是一个状态-动作对。这个回溯规则更新每个状态-动作对，为了使图 10.7b 的顶部的状态，回溯的根结点，为一个动作结点与产生它的状态为一对。

在 Q 学习中，回溯从动作结点开始，最大化下一个状态的所有可能动作和它们的奖励。在完全递归定义的 Q 学习中，回溯树的底部结点一个从根结点开始的动作和它们的后继动作的奖励的序列可以到达的所有终端结点。联机的 Q 学习，从可能的动作向前扩展，不需要建立一个完全的世界模型。Q 学习还可以脱机执行。我们可以看到，Q 学习是一种时序差分的方法。

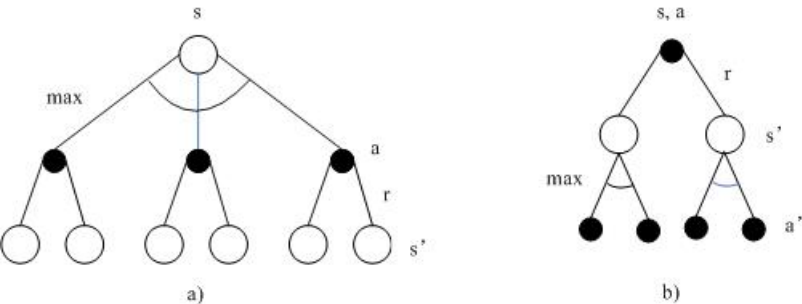


图 10.7 对 a)V*和 b)Q*的回溯图

Monte Carlo 方法采用一次学习循环所获得的整个返回函数去逼近实际的值函数，而强化学习方法使用下一状态的值函数（即 Bootstrapping 方法）和当前获得的瞬时奖励来估计当前状态值函数。显然，强化学习方法将需要更多次学习循环才能逼近实际的值函数。因此可以修改（10.8）式，构造一个新的 λ -返回函数 R'_t ，如式（10.16），其中假定系统在此次学习循环中第 T 步后进入终结状态。 λ -返回函数 R'_t 的物理意义如图 10.8 所示。那么值函数迭代即遵循式（10.17）。

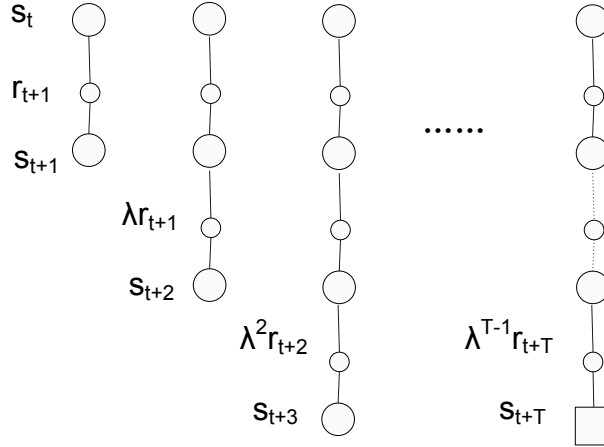


图 10.8 λ -返回函数

$$R'_t = r_{t+1} + \lambda r_{t+2} + \lambda^2 r_{t+3} + \cdots + \lambda^{T-1} r_{t+T} \quad (10.16)$$

$$V(s_t) \leftarrow V(s_t) + \alpha [R'_t - V(s_t)] \quad (10.17)$$

由于强化学习算法中值函数的更新是在每一学习步（即每次获得 $\langle s, a, r, s' \rangle$ 经验后）进行的，因此为使学习算法能在一次学习循环中值函数满足（10.17）式，设计新的 TD(λ) 算法。在 TD(λ) 算法中通过构造 $e(s)$ 函数，即可以保证在一次学习循环中值函数以（10.17）式更新。

算法 10.2 TD(λ) 算法

Initialize $V(s)$ arbitrarily and $e(s)=0$ for all $s \in S$

Repeat (for each episode)

Initialize s

Repeat (for each step of episode)

$a \leftarrow$ action given by π for s (e.g., ϵ -greedy)

Take action a , observe r, s'

$\delta \leftarrow r + \gamma V(s') - V(s)$

$e(s) \leftarrow e(s) + \delta$

for all s

$V(s) \leftarrow V(s) + \alpha \delta e(s)$

$e(s) \leftarrow \gamma \lambda e(s)$

$s \leftarrow s'$

Until s is terminal

我们可以将值函数的估计和策略评估两步骤合二为一。在算法中构造状态-动作对值函数，

即 Q 函数。Q 函数定义如(10.18)式。理论证明，当学习率 α 满足一定条件, Q 学习算法必然收敛于最优状态-动作对值函数[Tsitsiklis 1994]。Q 学习算法是目前最普遍使用的强化学习算法之一。

$$Q^\pi(s, a) = \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V^\pi(s')] \quad (10.18)$$

算法 10.3 Q 学习算法

```

Initialize  $Q(s, a)$  arbitrarily
Repeat (for each episode)
  Initialize  $s$ 
  Repeat (for each step of episode)
    Choose  $a$  from  $s$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
    Take action  $a$ , observe  $r, s'$ 
     $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$ 
     $s \leftarrow s'$ 
  Until  $s$  is terminal

```

10.7 强化学习中的函数估计

对于大规模 MDP 或连续空间 MDP 问题中，强化学习不可能遍历所有状态。因此要求强化学习的值函数具有一定泛化能力。强化学习中的映射关系包括： $S \rightarrow A$ 、 $S \rightarrow R$ 、 $S \times A \rightarrow R$ 、 $S \times A \rightarrow S$ 等等。强化学习中的函数估计本质就是用参数化的函数逼近这些映射。

用算子 Γ 来表示 (10.8) 式。假设初始的值函数记为 V_0 ，则学习过程产生的值函数逼近序列为：

$$V_0, \Gamma(V_0), \Gamma(\Gamma(V_0)), \Gamma(\Gamma(\Gamma(V_0))), \dots$$

在经典的强化学习算法中，值函数采用查找表 (lookup-table) 方式保存。而在函数估计中，采用参数化的拟合函数替代查找表。此时，强化学习基本结构如图 10.9 所示。记函数估计中 V 为目标函数， V' 为估计函数，则 $M: V \rightarrow V'$ 为函数估计算子。假设值函数初值为 V_0 ，则学习过程中产生的值函数序列为：

$$V_0, M(V_0), \Gamma(M(V_0)), M(\Gamma(M(V_0))), \Gamma(M(\Gamma(M(V_0)))) , \dots$$

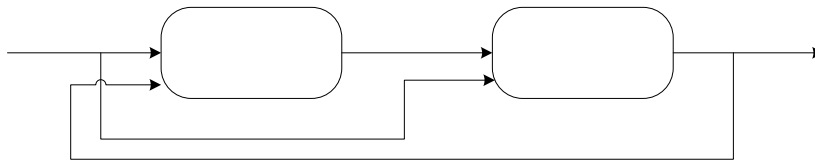


图 10.9 有函数估计的强化学习结构

因此，类似与 Q 学习算法，使用函数估计的强化学习算法迭代公式做以下修改。

$$Q(s, a) \leftarrow (1 - \alpha) V'(s, a) + \alpha (r(s, a, s') + \max_{a'} V'(s', a')) \quad (10.19)$$

$$V'(s, a) = M(Q(s, a)) \quad (10.20)$$

在函数估计强化学习中，同时并行两个迭代过程：一是值函数迭代过程 Γ ，另一是值函数逼近过程 M 。因此 M 过程逼近的正确性和速度都将对强化学习产生根本的影响。目前函数估计的方法通常采用有导师监督学习方法：如状态聚类[Singh 1995][Moore 1994]、函数插值[Davies 1997]和人工神经网络[Sutton 1996]等方法。

状态聚类将整个状态空间分成若干区域，在同一区域的状态认为其值函数相等。于是一个连续或较大规模的 MDP 问题被离散化为规模较小的 MDP 问题。状态聚类最简单的方法是区格法，它将状态空间的每一维等分为若干区间，而将整个状态空间划分为若干相同大小的区间，对二维来说就是区格划分。更复杂的划分方法是变步长划分和三角划分，采用状态聚类方法的函数估计强化学习已经被证明是收敛的。需要指明的是：尽管状态聚类强化学习是收敛的，但其并不一定收敛到原问题的最优解上。要使收敛的值函数达到一定的精度，状态聚类的步长不能太大。因此对于大规模 MDP 问题，它仍然面临着维数灾难的困难。

目前函数估计强化学习研究的热点是采用神经网络等方法进行函数估计。但尽管这些新方法可以大幅度提高强化学习的学习速度，但并不能够保证收敛性。因此研究既能保证收敛性，又能提高收敛速度的新型函数估计方法，仍然是目前函数估计强化学习研究的重点之一。

10.8 强化学习的应用

在马尔科夫决策过程中，主体可感知到其环境的不同状态集合，并且有它可执行的动作集合。在每个离散时间步 t ，主体感知到当前状态 s_t ，选择当前动作 a_t 并执行它。环境响应给出奖励 $r_t = Q(s_t, a_t)$ ，并产生一个后继状态 $s_{t+1} = P(s_t, a_t)$ 。在马尔科夫决策过程中，其中函数 $Q(s_t, a_t)$ 称之为动作评估函数（价值函数）， $P(s_t, a_t)$ 称之为状态转换函数；其中 $Q(s_t, a_t)$ 和 $P(s_t, a_t)$ 只依赖于当前状态和动作，而不依赖于以前的状态和动作；而强化学习正是处理 MDP 的一种重要方法。强化学习通过学习动作评估函数 $Q(s_t, a_t)$ 或状态转换函数 $P(s_t, a_t)$ 来达到学习的目的。而 Q 学习主要是通过学习 $Q(s_t, a_t)$ 获得最大的奖励。

这里应用 Q 学习算法进行仿真机器人足球 2 对 1 训练，训练的目的在于试图使主体学习获得一种战略上的意识，能够在进攻中进行配合[宋志伟, 2003]。如图 10.10，前锋 A 控球，并且在可射门的区域内，但是 A 已经没有射门角度了；队友 B 也处于射门区域，并且 B 具有良好的射门角度。A 传球给 B，射门由 B 来完成，那么这次进攻配合就会很成功。通过 Q 学习的方法来进行 2 对 1 的射门训练，让 A 掌握在这种状态情况下传球给 B 的动作是最优的策略；主体通过大量的学习训练（大数量级的状态量和重复相同状态）来获得策略，因此更具有适应性。

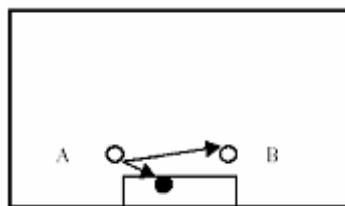


图 10.10 仿真机器人足球 2 对 1 训练

图 10.11 给出状态描述，将进攻禁区划分为 20×8 个小区域，每个小区域是边长为 2m 的正方形，一个二维数组 $A_{i,j}$ ($0 \leq i \leq 19, 0 \leq j \leq 7$) 便可描述这个区域。使用三个 Agent 的位置来描述 2 对 1 进攻时的环境状态，利用图 10.11 所示的划分来泛化状态。可认为主体位于同一战略区域为相似状态，这样对状态的描述虽然不精确，但设计所需的是一种战略层次的描述，可认为 Agent 在战略区域内是积极跑动的，这种方法满足了需求。如此， $\langle S_A, S_B, S_G \rangle$ 便描述了一个特定的状态；其中， S_A 是进攻队员 A 的区域编号， S_B 是进攻队员 B 的区域编号， S_G 是守门员的区域编号。区域编号计算公式为： $S = i \times 8 + j$ 。相应的，所保存的状态值为三个区域编号组成的对。

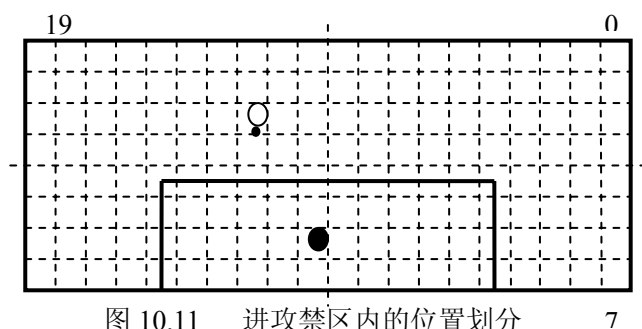


图 10.11 进攻禁区内的位置划分

可选动作集确定为 $\{Shoot, Pass, Dribble\}$ 。

Shoot 的策略通过基于概率的射门训练的学习来得到。

Dribble 的策略是，始终向受到威胁小，并且射门成功率高的区域带球。为了实现这一策略目标，可划分进攻区域为多个战略区，在每个战略区进行射门评价，记录每个区域的射门成功率。

Pass 策略很简单，只需在两个 Agent 间进行传球，即不需要选择球传送的对象，也不需要判断传球路径。如果传球失败的话，则认为在这种状态下执行 *Pass* 策略是不成功的；经过此训练后，不可能的传球路径也不会被执行了。

训练中的所有状态包含了四个吸收状态。假设进攻方在左半场，按照标准的 Soccer server 规范，这四个状态的比赛模式为 *play_on*、*goal_left*、*goal_kick_right* 和 *free_kick_right*。当达到吸收状态时，给与主体最终奖励 r 。促使到达吸收状态的上一步动作获得的立即回报值为最

终奖励值 r ，其他未直接促使到达吸收状态的动作均获得过程奖励值作为立即奖励；其中 `goal_left` 的 r 最大为 1 表示进球，其他状态下 r 为不同大小的负数。

主体在经过一定的状态和执行多个动作后获得了终态奖励（到达了吸收状态），这时就会对这个状态-动作序列分配奖励。Q 学习算法的核心就是每一个状态和动作的组合都拥有一个 Q 值，每次获得最终回报后通过更新等式更新这个 Q 值。由于 Robocup 仿真平台在设计的时候在状态转换的时候加入了一个较小的随机噪音，所以该模型为非确定 MDP，确定 Q 更新等式为：

$$Q(s, a) = (1 - \alpha)Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s_{t+1}, a')) \quad (10.21)$$

规定 $\alpha=0.1$ ， $\gamma=0.95$ 。

在实际的训练中，初始 Q 表各项的值为 1。经过大约 2 万次的训练（到达吸收状态为一次），Agent 的 Q 表中的绝大部分项发生了变化，并且已经区分开来。表 10.1 是某场景下的训练次数和动作选择的变化示意表。

表 10.1 Q 值变化表

	初始值	5千次	1万次	2万次
shoot	1	0.7342	0.6248	0.5311
pass	1	0.9743	0.9851	0.993
dribble	1	0.9012	0.8104	0.7242

强化学习的应用主要可为制造过程控制、各种任务调度、机器人设计和游戏等。在过去的二十年中，强化学习研究取得了突破性进展，但目前仍然存在许多有待解决的问题。强化学习的一个主要缺点是收敛慢。其根本原因在于学习过程仅仅从经验获得的奖励中进行策略的改进，而忽略了大量其他有用的领域信息。因此，如何结合其他机器学习技术，如神经网络、符号学习等技术，来帮助系统加快学习速度是强化学习研究和应用的重要方向。目前，结合技术研究的主要难点在于：如何从理论上证明和保证学习算法的收敛性。在更复杂马尔氏决策模型中发展有效的强化学习算法也将是未来重要的研究方向之一。

习 题

- 1 简述强化学习的主要分支和研究历程。
- 2 试解释强化学习模型及其与其他机器学习方法的异同。
- 3 试解释马尔可夫决策过程和及其本质。
- 4 简述蒙特卡罗方法的基本思想及其在强化学习中的应用。
- 5 说明时序差分学习的基本思想并以九宫游戏为例说明时序差分学习方法的执行过程。
- 6 考虑下图显示的一个确定性格子世界，其中含有吸收目标 G(目标状态)。图中做了标记的为立即回报，而其他没有做标记的转换都为 0。给出格子世界的每个状态的最大折算累积

回报 V^* ，给出每个转换的最大折算累积回报 $Q(s,a)$ 的值。并写出一个最优策略，使用 $\gamma = 0.8$ 。

第十一章 粗糙集

11.1 概 述

在经典逻辑中，只有真、假值之分。但在现实生活中有许多含糊现象并不能简单地用真、假值来表示，如何表示和处理这些现象就成为一个研究领域。长期以来许多逻辑学家和哲学家就致力于研究含糊概念。早在 1904 年谓词逻辑的创始人 G. Frege 就提出了含糊(Vague)一词，他把它归结到边界线上，也就是说在全域上存在一些个体既不能在其某个子集上分类，也不能在该子集的补集上分类。

1965 年，Zadeh 提出了模糊集，不少理论计算机科学家和逻辑学家试图通过这一理论解决 G. Frege 的含糊概念，但遗憾的是模糊集是不可计算的，即没有给出数学公式描述这一含糊概念，故无法计算出它的具体的含糊元素数目，如模糊集中的隶属函数 μ 和模糊逻辑中的算子 λ 都是如此。时隔 20 年后的 80 年代初，波兰的 Pawlak 针对 G. Frege 的边界线区域思想提出了粗糙集(Rough Set) [Pawlak 82]，他把那些无法确认的个体都归属于边界线区域，而这种边界线区域被定义为上近似集和下近似集之差集。由于它有确定的数学公式描述，所以含糊元素数目是可以计算的，即在真假二值之间的含糊度是可以计算的。粗糙集理论主要兴趣在于它恰好反映了人们用粗糙集方法处理不分明问题的常规性，即以不完全信息或知识去处理一些不分明现象的能力，或依据观察、度量到的某些不精确的结果而进行分类数据的能力。八十年代以来经过许多计算机科学家和数学家的不懈研究，已经从理论上日趋完善，特别是由于八十年代末和九十年代初在知识发现等领域得到了成功的应用而越来越受到国际上的广泛关注。相对于其他处理不确定性和模糊性的理论工具而言，粗糙集理论有着许多不可替代的优越性。经过近几年的研究和发展，它已经在信息系统分析、人工智能及应用、决策支持系统、知识与数据发现、模式识别与分类、故障检测等方面取得了较为成功的应用。

粗糙集概念在某种程度上与许多其它为处理含糊和不精确性问题而研制的数学工具有相似之处，特别是和 Dempster-Shafer 证据理论 [Shafer 76]。两者之间主要区别在于 Dempster-Shafer 理论利用信度函数作为主要工具，而粗糙集理论利用集合：下近似集和上近似集。另一种关系存在于模糊集理论和粗糙集理论之间。粗糙集理论与模糊集理论多方面对照，不是和模糊集竞争，而是补充它。总之，粗糙理论和模糊集理论对于不完全的知识来说它们是各自独立的方法。此外，有一些关系存在于粗糙理论与辨别式分析之间，与 Boolean 推理方法之间、与决策分析之间。

粗糙理论的主要优势之一是它不需要任何预备的或额外的有关数据信息，比如统计学中的概率分布，Dempster-Shafer 理论中的基本概率赋值，或者模糊集理论中的隶属度或概率值。

八十年代，许多波兰学者对粗糙集理论及其应用进行了坚持不懈的深入研究，其中对粗糙集理论的数学性质与逻辑系统进行了广泛的分析。当时大多数研究成果发表在“*Bulletin of The Polish Academic of Science: Mathematics*”或“*Bulletin of The Polish Academic of Science: Technical Science*”上；同时，他们也开发了一些应用系统。

1991 年波兰 Z. Pawlak 教授的第一本关于粗糙集的专著和 1992 年 R. Slowinski 主编的关于粗糙集应用及其与相关方法比较研究的论文集的出版，推动了国际上对粗糙集理论与应用的深入研究。1992 年在波兰 Kiekrz 召开了第 1 届国际粗糙集讨论会。这次会议着重讨论了集合近似定义的基本思想及其应用，其中粗糙集环境下机器学习的基础研究是这次会议的四个专题之

一。但是，参加这次会议的研究者较少，范围也不太广泛。这次会议选出 15 篇论文刊登在“*Foundation of Computing and Decision Sciences*”1993 年第 18 卷上。从此每年召开一次与粗糙集理论为主题的国际研讨会。

1993 年在加拿大 Banff 召开了第二届国际粗糙集与知识发现(RSD'93)研讨会。这次会议极大地推动了国际上对粗糙集理论与应用的研究，其主题是粗糙集、Fuzzy 集与知识发现。由于此时正值 KDD 成为研究的热门话题，一些著名 KDD 学者参加这次会议，并且介绍了许多基于扩展的粗糙集理论的知识发现方法与系统。1994 年在美国的 San Jose 召开了第 3 届国际粗糙集与软计算研讨会，这次会议广泛地探讨了粗糙集与模糊逻辑、神经网络、进化理论等的融合问题。

粗糙集理论及应用的几位主要倡导者，在 1995 年第 11 期 ACM 通讯上撰文，概括性地介绍了目前人工智能应用新技术之一的粗糙集理论的基本概念，及其在知识获取和机器学习、决策分析、知识发现等领域的具体研究项目和进展。1995 年在美国 Willmington 召开了粗糙集研讨会。特别值得一提的是 1995 年召开的第 4 届模糊理论与技术国际研讨会(Fuzzy Theory & Technology'95)，在这次会议上，针对粗糙集与模糊集合的基本观点与相互关系展开了激烈的讨论，较大地促进了粗糙集的研究。1996 年底在日本东京召开了第 5 届国际粗糙集研讨会，这是第一次在亚洲地区召开的范围广泛的粗糙集研讨会。1998 年 6 月在波兰华沙召开了“第一届粗糙集和计算的当前趋势”学术会议。1999 年 11 月 9—11 日在日本召开了“第七届粗糙集、Fuzzy 集、数据挖掘和粒度—软计算的国际学术研讨会”(The Seventh International Workshop on Rough Sets, Fuzzy Sets, Data Mining and Granular-Soft Computing (RSFDGRC'99))，阐述了当前粗糙集、模糊集的研究现状和发展趋势，指出将着重在软计算、数据库、AI 和近似推理等理论和应用方面发展。

2000 年 10 月在加拿大又召开了“第二届粗糙集和计算的当前趋势”学术会议。目前，在许多关于人工智能、模糊理论、信息管理与知识发现等国际学术会议上经常可以看到许多涉及粗糙集的论文。

必须指出，粗糙理论也不是万能的，对建模而言，尽管粗糙理论对知识不完全的处理是有效的，但是，由于这个理论未包含处理不精确或不确定原始数据的机制。因此，单纯地使用这个理论不一定能有效地描述不精确或不确定地实际问题，这意味着，需要其它方法补充。一般地说，由于证据理论与模糊集理论等具有处理不精确和不确定数据的方法（尽管在描述上不一定方便），因此，将它们与粗糙理论构成互补是自然的考虑。

为了能更好地理解粗糙集理论的本质和特点，这里介绍粗糙集理论的一些基本定义，目的在于阐明粗糙集的思想本质，以及与其他处理不确定性和模糊性数学工具的不同之处。

11.1.1 知识的分类观点

基本粗糙集理论认为知识就是人类和其他物种所固有的分类能力。例如，在现实世界中关于环境的知识主要表明了生物根据其生存观来对各种各样的情形进行分类区别的能力。每种生物根据其传感器信号形成复杂的分类模式，就是这种生物的基本机制。分类是推理、学习与决策中的关键问题。因此，粗糙集理论假定知识是一种对对象进行分类的能力。这里的“对象”是指我们所能言及的任何事物，比如实物、状态、抽象概念、过程和时刻等等。即知识必须与具体或抽象世界的特定部分相关的各种分类模式联系在一起，这种特定部分称之为所讨论的**全域或论域(universe)**。对于全域及知识的特性并没有任何特别假设。事实上，知识构成了某一感兴趣领域中各种分类模式的一个族集(family)，这个族集提供了关于现实的显事实，以及能够从这些显事实中推导出隐事实的推理能力。

为数学处理方便起见，在下面的定义中用等价关系来代替分类。

定义 11.1 一个近似空间(approximate space) (或知识库) 定义为一个关系系统 (或二元组)

$$K=(U, R), \quad (11.1)$$

其中 $U \neq \emptyset$ (\emptyset 为空集) 是一个被称为全域或论域(universe)的所有要讨论的个体的集合, R 是 U 上等价关系的一个族集。

定义 11.2 设 $P \subseteq R$, 且 $P \neq \emptyset$, P 中所有等价关系的交集称为 P 上的一种不分明关系 (indiscernibility relation) (或称不可区分关系), 记作 $IND(P)$, 即

$$[x]_{IND(P)} = \bigcap_{R \in P} [x]_R \quad (11.2)$$

注意, $IND(P)$ 也是等价关系且是唯一的。

定义 11.3 给定近似空间 $K=(U, R)$, 子集 $X \subseteq U$ 称为 U 上的一个概念(concept), 形式上, 空集 \emptyset 也视为一个概念; 非空子族集 $P \subseteq R$ 所产生的不分明关系 $IND(P)$ 的所有等价类关系的集合即 $U/IND(P)$, 称为基本知识(basic knowledge), 相应的等价类称为基本概念(basic concept); 特别地, 若关系 $Q \in R$, 则关系 Q 就称为初等知识(elementary knowledge), 相应的等价类就称为初等概念(elementary concept)。

一般用大写字母 P, Q, R 等表示一个关系, 用大写黑体字母 $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ 等表示关系的族集; $[x]_R$ 或 $R(x)$ 表示关系 R 中包含元素 $x \in U$ 的概念或等价类。为了简便起见, 有时用 \mathbf{P} 代替 $IND(P)$ 。根据上述定义可知, 概念即对象的集合, 概念的族集 (分类)就是 U 上的知识, U 上分类的族集可以认为是 U 上的一个知识库, 或说知识库即是分类方法的集合。

11.1.2 新型的隶属关系

粗糙集理论与传统的集合理论有着相似之处, 但是它们的出发点完全不同。传统集合论认为, 一个集合完全是由其元素所决定, 一个元素要么属于这个集合, 要么不属于这个集合, 即它的隶属函数 $\mu_X(x) \in \{0, 1\}$ 。模糊集合对此做了拓广, 它给成员赋予一个隶属度, 即 $\mu_X(x) \in [0, 1]$, 使得模糊集合能够处理一定的模糊和不确定数据, 但是其模糊隶属度的确定往往具有人为因素, 这给其应用带来了一定的不便。而且, 传统集合论和模糊集合论都是把隶属关系作为原始概念来处理, 集合的并和交就建立在其元素的隶属度 \max 和 \min 操作上, 因此其隶属度必须事先给定 (传统集合默认隶属度为 1 或 0)。在粗糙集中, 隶属关系不再是一个原始概念, 因此无需人为给元素指定一个隶属度, 从而避免了主观因素的影响。而且认为不确定与隶属关系有关, 而模糊性则表现在集合本身。为描述不确定性, 定义隶属关系如下。

定义 11.4 设 $X \subseteq U$ 且 $x \in U$, 集合 X 的隶属函数(membership function) (或称为粗糙隶属函数(rough membership function)) 定义为

$$\mu_X^R(x) = \frac{\text{card}(X \cap R(x))}{\text{card}(R(x))} \quad (11.3)$$

其中 R 是不可区分关系, $R(x)=[x]_R=\{y:(y \in U) \wedge (y R x)\}$ 。

在本文中我们用 card 表示集合的基数。

根据上面的定义, 可以得到以下性质:

- 1) $\mu_X^R(x)=1$ 当且仅当 $[x]_R \subseteq X$;
- 2) $\mu_X^R(x)>0$ 当且仅当 $[x]_R \cap X \neq \emptyset$;

3) $\mu_X^R(x)=0$ 当且仅当 $[x]_R \cap X = \emptyset$ 。

显然有 $\mu_X^R(x) \in [0,1]$ 。我们可以看到，这里的隶属关系是根据已有的分类知识客观计算出来的，可以被解释为一种条件概率，能够从全域上的个体加以计算，而不是主观给定的。

11.1.3 概念的边界观点

知识的粒度性是造成使用已有知识不能精确地表示某些概念的原因。这就产生了所谓的关于不精确的“边界”思想。著名哲学家 Frege 认为“概念必须有明确的边界。没有明确边界的概念，将对应于一个在周围没有明确界线的区域”。粗糙集理论中的模糊性就是一种基于边界的概念，即一个不精确的概念具有模糊的不可被明确划分的边界。为刻画模糊性，每个不精确概念由一对称为上近似与下近似的精确概念来表示，它们可用隶属函数定义如下。

定义 11.5 设集合 $X \subseteq U$ ，我们称 $R_*(X) = \{x: (x \in U) \wedge (\mu_X^R(x) = 1)\}$ 为 X 的下近似(lower approximation)， $R^*(X) = \{x: (x \in U) \wedge (\mu_X^R(x) > 0)\}$ 为 X 的上近似(upper approximation)， $BN_R(X) = R^*(X) - R_*(X)$ 为 X 的边界或边界区域(boundary)。

显然，若 $BN_R(X) \neq \emptyset$ 或 $R^*(X) \neq R_*(X)$ ，则集合 X 就是一个粗糙概念。下近似包含了所有使用知识 R 可确切分类到 X 的元素，上近似则包含了所有那些可能是属于 X 的元素。上近似与下近似的差就是此概念的边界区域，它由不能肯定分类到这个概念或其补集中的所有元素组成。在粗糙集中也采用下面的说法， $POS_R(X) = R_*(X)$ 称为集合 X 的 R -正区域(R -positive region)， $NEG_R(X) = U - R^*(X)$ 称为集合 X 的 R -反区域(R -negative region)。

我们可以看到，模糊性和不确定性在此有了联系，即模糊性是由不确定性来定义的。一般地，在给定近似空间中，并非所有的对象子集都可用给定的知识来表示成概念，这样的子集就认为是粗糙概念（即不精确的或近似的概念）。但是，粗糙概念可以通过两个精确概念（上近似和下近似）来粗糙地定义，这就使我们可以精确地讲述不精确的概念。另外，用隶属函数、上近似和下近似还可以定义粗糙集的包含关系与相等关系等。

11.2 知识的约简

本节着重叙述粗糙集理论的两个最基本问题：知识约简与知识的依赖性。知识约简是研究近似空间中每个等价关系是否都是必要的，以及如何删去不必要的知识。知识约简在信息系统分析与数据挖掘等领域都具有重要的应用意义。知识之间的依赖性决定知识是否可以进行约简，根据依赖性所定义的知识的重要性往往是知识约简的重要启发式信息。

11.2.1 一般约简

在粗糙集理论应用中，约简与核是两个最重要的基本概念。直观的，所谓知识的约简是指知识的本质部分，它足以定义所考虑的知识中遇到的所有基本概念，而核是其最重要的部分。

定义 11.6 设 R 是等价关系的一个族集，且设 $R \in R$ 。若 $IND(R) = IND(R - R)$ ，则称关系 R 在族集 R 之中是可省的(dispensable)，否则就是不可省的。若族集 R 中的每个关系 R 都是不可省的，则称族集 R 是独立的(independent)，否则就是依赖的或非独立的。

定义 11.7 若 $Q \subseteq P$ 是独立的，并且 $IND(Q) = IND(P)$ ，则称 Q 是关系族集 P 的一个约简

(reduct)。在族集 \mathbf{P} 中所有不可省的关系的集合称为 \mathbf{P} 的核(core)，以 $CORE(\mathbf{P})$ 来表示。

显然，族集 \mathbf{P} 有多个约简（约简的不唯一性）。

下面的定理是建立核与约简之间联系的重要性质。

定理 11.1 族集 \mathbf{P} 的核等于 \mathbf{P} 的所有约简的交集。即

$$CORE(\mathbf{P}) = \cap RED(\mathbf{P}) \quad (11.4)$$

其中 $RED(\mathbf{P})$ 是 \mathbf{P} 的所有约简的族集。

从上面定理可看出，核的概念具有两方面意义。首先，可作为计算所有约简的基础，因为核包含于每一个约简之中，并且其计算是直接的。其次，核可以解释为知识最重要部分的集合，进行知识约简时不能够删除它。

一般产生约简的方法是逐个向核中添加可省的关系，并进行检查。注意，可省的关系集合的幂集的基数是多少，就有多少种添加的方式。最好的情况是所有不可省的关系集合本身就是约简，此时的约简是唯一的。所以，计算所有约简与计算一个最佳约简（比如定义为关系最少）都是 NP 难题。

11.2.2 相对约简

本节推广约简与核的概念。为此，首先需要定义一个分类关于另一个分类的正区域概念。

定义 11.8 设 \mathbf{P} 和 \mathbf{Q} 是全域 U 上的等价关系的族集，所谓族集 \mathbf{Q} 的 \mathbf{P} -正区域(P-positive region of \mathbf{Q})，记作 $POS_{\mathbf{P}}(\mathbf{Q})$ ，定义为

$$POS_{\mathbf{P}}(\mathbf{Q}) = \bigcup_{X \in U/\mathbf{Q}} \mathbf{P}_*(X) \quad (11.5)$$

族集 \mathbf{Q} 的 \mathbf{P} -正区域是全域 U 的所有那些使用分类 U/\mathbf{P} 所表达的知识，能够正确地分类于 U/\mathbf{Q} 的等价类之中的对象的集合。一个集合 X 相对于一个等价关系 \mathbf{P} 的正区域就是这个集合的下近似 $\mathbf{P}_*(X)$ ；而一个等价关系 \mathbf{Q} 相对于另一个等价关系 \mathbf{P} 的正区域的概念是解决分类 \mathbf{Q} 的等价类(一般视为决策类)之中的那些对象可由分类 \mathbf{P} 的等价类(一般视为条件类)来分类的问题。

定义 11.9 设 \mathbf{P} 和 \mathbf{Q} 是全域 U 上的等价关系的族集， $R \in \mathbf{P}$ 。若

$$POS_{IND(\mathbf{P})}(IND(\mathbf{Q})) = POS_{IND(\mathbf{P}-\{R\})}(IND(\mathbf{Q})) \quad (11.6)$$

则称关系 R 在族集 \mathbf{P} 中是 \mathbf{Q} -可省的，否则称为 \mathbf{Q} -不可省的；如果在族集 \mathbf{P} 中的每个关系 R 都是 \mathbf{Q} -不可省的，则称 \mathbf{P} 关于 \mathbf{Q} 是独立的，否则就称为是依赖的。

定义 11.10 $S \subseteq \mathbf{P}$ 称为 \mathbf{P} 的 \mathbf{Q} -约简(Q-reduct)，当且仅当 S 是 \mathbf{P} 的 \mathbf{Q} -独立的子族集，且 $POS_S(\mathbf{Q}) = POS_{\mathbf{P}}(\mathbf{Q})$ ；族集 \mathbf{P} 中的所有 \mathbf{Q} -不可省的初等关系的集合，称为族集 \mathbf{P} 的 \mathbf{Q} -核(Q-core)，记作 $CORE_{\mathbf{Q}}(\mathbf{P})$ 。

容易看出，当 $\mathbf{P} = \mathbf{Q}$ 时，上述定义就是 11.2.1 节所引入的定义。

下面的定理是定理 11.1 的拓广。

定理 11.2 族集 \mathbf{P} 的 \mathbf{Q} -核等于族集 \mathbf{P} 的所有 \mathbf{Q} -约简的交集。即

$$CORE_{\mathbf{Q}}(\mathbf{P}) = \cap RED_{\mathbf{Q}}(\mathbf{P}) \quad (11.7)$$

其中 $RED_{\mathbf{Q}}(\mathbf{P})$ 是族集 \mathbf{P} 的所有 \mathbf{Q} -约简的族集。

假设 \mathbf{P} 与 \mathbf{Q} 是全域 U 上的等价关系的族集（知识），族集 \mathbf{Q} 的族集 \mathbf{P} -正区域 $POS_{\mathbf{P}}(\mathbf{Q})$ 是使用知识 \mathbf{P} 能够分类于知识 \mathbf{Q} 的概念之中的所有对象的集合。如果整个知识 \mathbf{P} 对于将对象分类于知识 \mathbf{Q} 的概念中都是必要的，那么知识 \mathbf{P} 就是 \mathbf{Q} -独立的，知识 \mathbf{P} 的 \mathbf{Q} -核知识是知识 \mathbf{P} 的本质部分，在不影响将对象分类于 \mathbf{Q} 的概念之中的能力的前提之下， \mathbf{Q} -核知识是不能被删除的。即删除它们的任何部分都将会影响知识 \mathbf{P} 把对象分类于 \mathbf{Q} 的概念之中的能力。知识 \mathbf{P} 的 \mathbf{Q} -约简是知识 \mathbf{P} 的某种最小子集，它具有与整个知识 \mathbf{P} 相同的把对象分类于知识 \mathbf{Q} 的概念之中的能力。注意，知识 \mathbf{P} 可能有多个约简。在某种意义上，如果知识 \mathbf{P} 仅仅只有一个 \mathbf{Q} -约简，

则知识 P 就是确定的，即把对象分类于知识 Q 的概念之中时，仅仅只有一种使用知识 P 的方式。当知识 P 为不确定的即知识 P 有多个 Q -约简，将对象分类于 Q 的概念时，一般就有多种使用知识 P 的方式。若核为空，则这种不确定性就尤其严重。

11.2.3 知识的依赖性

知识的依赖性可形式定义如下：

定义 11.11 设 $K=(U, R)$ 是一个近似空间， $P, Q \subseteq R$ 。

- 1) 知识 Q 依赖于知识 P 或知识 P 可推导出知识 Q ，当且仅当 $IND(P) \subseteq IND(Q)$ ，记作 $P \Rightarrow Q$ ；
- 2) 知识 P 和知识 Q 是等价的，当且仅当 $P \Rightarrow Q$ 且 $Q \Rightarrow P$ ，即 $IND(P) = IND(Q)$ ，记作 $P = Q$ ，明显地， $P = Q$ 当且仅当 $IND(P) = IND(Q)$ ；
- 3) 知识 P 和知识 Q 是独立的，当且仅当 $P \Rightarrow Q$ 且 $Q \Rightarrow P$ 均不成立，记作 $P \neq Q$ 。

依赖性也可以是部分成立的，也就是从知识 P 能推导出知识 Q 的一部分知识，或者说知识 Q 只有一部分依赖于知识 P 的。部分依赖性（部分可推导性）可以由知识的正区域来定义。现在我们形式地定义部分依赖性。

定义 11.12 设 $K=(U, R)$ 是一个知识库， $P, Q \subseteq R$ ，我们称知识 Q 以依赖度 $k(0 \leq k \leq 1)$ 依赖于知识 P ，记作 $P \Rightarrow_k Q$ ，当且仅当

$$k = \gamma_P(Q) = \text{card}(POS_P(Q)) / \text{card}(U) \quad (11.8)$$

- (1) 若 $k=1$ ，则称知识 Q 完全依赖于知识 P ， $P \Rightarrow_1 Q$ 也记成 $P \Rightarrow Q$ ；
- (2) 若 $0 < k < 1$ ，则称知识 Q 部分依赖于知识 P ；
- (3) 若 $k=0$ ，则称知识 Q 完全独立于与知识 P 。

上述思想也可解释为将对象分类的能力。确切地说，若 $k=1$ ，则全域的所有元素都能够用知识 P 来分类于 U/Q 的概念之中。若 $k \neq 1$ ，则仅仅是全域之中属于正区域的那些元素能够用知识 P 来分类于 U/Q 的概念之中。特别地，若 $k=0$ ，则全域的所有元素都不能够用知识 P 来分类于 U/Q 的概念之中。因此，系数 $\gamma_P(Q)$ 可以理解为知识 P 和知识 Q 之间的依赖程度。

11.3 决策逻辑

粗糙集理论中近似空间的概念给出了一种理解与处理知识的基本框架，从算法实现角度考虑，它特别适合于处理以数据表形式表示的信息，而这样的数据表称为信息系统或知识表达系统。这种表达方式有很大的优点，比如它的直观性，但我们也可以从不同的角度来看待这种数据表，也就是说，它可以被看作是关于现实和结果的一个命题的集合，因此用逻辑工具来处理，这就是本节的出发点。

11.3.1 决策表的公式化定义

知识表达系统可以公式化地定义为： $S=(U, A)$ ，其中 U 为非空的有限集论域， A 为非空的属性有限集。

知识表达系统这个定义，我们可以方便地用表格表达来实现。知识的表格表达法可以看作是一种特殊的形式语言，它用符号来表达等价关系。在知识表达系统数据表中，列表示属性，行表示对象（如状态、过程等），并且每行表示该对象的一条信息。数据表可以通过观察、测量得到。

容易看出，一个属性对应一个等价关系，一个表可以看作是定义的一族等价关系，即知识

库。前面所讨论的知识约简就可以转化为属性约简。

对于每个属性子集 $B \subseteq A$ ，我们可以定义一个不可分辨二元关系 $IND(B)$ ，即

$$IND(B) = \{(x, y) \in U^2, \text{ 对于每一 } b \in B, b(x) = b(y)\}, \quad (11.9)$$

显然 $IND(B)$ 是一个等价关系，且 $b \in B, IND(B) = \cap IND(b)$ 。

每个子集 $B \subseteq A$ 称为一个属性，当 B 中只有一个元素时，则称 B 为原始的，否则称为复合的。属性 B 可以看作作用等价关系表示的知识的一个名称，可称为标识属性。包含对象 x 、具有属性 $B \subseteq A$ 的初等范畴的名称是一个集合对（属性、值），记为 $\{b, b(x)\}_{b \in B}$ 。

因知识库与知识表达系统之间具有一对一映照关系，取决于属性和属性名称的同构，故任一知识库 $K=(U, R)$ 、一个知识表达系统 $S=(U, A)$ ，两者之间的关系为：当 $r \in R$ 且 $U/r = \{x_1, x_2, \dots, x_n\}$ ，对于属性集 A ，每个属性 $a_r: U \Rightarrow V_{a_r}$ ，当且仅当 $x \in X_i, (i=1, 2, \dots, k)$ ，有 $V_{a_r} = \{1, 2, \dots, k\}$ ，

且 $a_r(x) = i$ 。这样，所有涉及知识库的定义都可以用知识表达系统的定义来描述。

因此，知识库中任一等价关系在知识系统数据表中表示为一个属性和用属性值表示的关系的等价类，表中的列可看作某些范畴的名称，而整个表包含了相应知识库中所有范畴的描述，包含了能从表中数据推导出的所有可能的规律。所以，数据表知识表达系统是对知识库中有效事实和规律的描述。

决策表是一类特殊而重要的知识表达系统，它指当满足某些条件时，决策（行为）应当怎样进行。多数决策问题都可以用决策表形式来表示，这一工具在决策应用中起着重要的作用。

决策表可以根据知识表达系统定义如下：

$S=(U, A)$ 为一知识表达系统，且 $C, D \subseteq A$ 是两个属性子集，分别称为条件属性和决策属性，具有条件属性和决策属性的知识表达系统可表示为决策表，记作 $T=(U, A, C, D)$ 或简称 CD 决策表。关系 $IND(C)$ 和关系 $IND(D)$ 的等价类分别称为条件类和决策类。

11.3.2 决策逻辑语言

我们把数据表看作是一种特殊逻辑模型，这里称为决策逻辑。它将用来从知识表达系统所获得的知识推导出结论，我们将使用逻辑推算的符号工具，发现知识的依赖性和进行知识的约简。

由于我们主要感兴趣的是决策逻辑，尽管通常的算法是指一系列的指令，这里我们仍然用决策算法这个术语来表示一个决策规则的集合。另外，普通公式可以为真或为假，在这里决策算法作为公式的集合没有为真为假的性质。决策算法作为公式的集合，决策算法的基本特点是一致或不一致，因而我们主要考虑数据的一致性，用计算的方法从数据中推导规则。

在这里我们定义和讨论的决策逻辑语言由原子公式组成，公式是一种属性-数据对，用命题联结词：与、或、非等，通过标准的方法构成复合公式。

首先我们定义决策逻辑语言的基本表达如下：

- (1) A ——属性常数集合；
- (2) $V = \cup V_a, a \in A, V$ ——属性值常数集合；
- (3) 命题联词的集合： $\{\sim, \vee, \wedge, \rightarrow, \equiv\}$ ，分别表示：否定、析取、合取、蕴含和等价。

我们注意到，命题联词符号可以看作逻辑联词：“非”、“或”、“与”、“当……，则……”、“当且仅当”。

基本表达不包含变量，其表达仅由属性和属性值符号、命题联词、辅助符号（如括号）等构成。

决策逻辑语言的公式集合是满足下列条件的最小集合：

- (1) 公式 (a, v) (或简写为 a_v) 成为初等原子公式, 是对于任何 $a \in A$ 且 $v \in V_a$ 的决策逻辑语言的公式;
- (2) 如果 θ 和 ψ 为决策逻辑语言的公式, 则 $\sim\theta, \theta \vee \psi, \theta \wedge \psi, \theta \rightarrow \psi, \theta \equiv \psi$ 也是决策逻辑语言的公式。

11.3.3 决策逻辑语言的语义

公式是用来描述论域中对象的工具, 可以用来描述论域中具有某些性质的对象的子集。例如在原子公式中, 有序对 (a, v) 解释为对属性 a 有值 v 的所有对象的描述。

为了准确地表达决策逻辑语言, 我们可以用模型和可满足性的概念来定义决策逻辑语言的 Tarski 式语义, 即使用模型来表示知识表达系统 $S=(U, A)$, 模型 S 描述论域 U 中谓词符号 (a, v) 的含义, 表达了某些对象的性质。利用可满足性的概念来定义决策逻辑语言, 即:

如果 S 是可理解的, 一个对象 $x \in U$ 满足 $S=(U, A)$ 中的公式 θ , 且记为 $x|_S \models \theta$ (简记为 $x| \models \theta$), 当且仅当满足下列条件时:

- (1) 当且仅当 $f(a, x)=v, x|=(a, v)$;
- (2) 当且仅当非 $x| \models \theta, x| \models \sim\theta$;
- (3) 当且仅当 $x| \models \theta$ 或 $x| \models \psi, x| \models \theta \vee \psi$;
- (4) 当且仅当 $x| \models \theta$ 和 $x| \models \psi, x| \models \theta \wedge \psi$;

作为上述条件的推论, 我们得到:

当且仅当 $x| \models \sim\theta \vee \psi, x| \models \theta \rightarrow \psi$;

当且仅当 $x| \models \theta \rightarrow \psi$ 和 $x| \models \psi \rightarrow \theta, x| \models \psi \equiv \theta$ 。

当 θ 为一公式时, 集合 $|\theta|_S$, 定义为:

$$|\theta|_S = \{x \in U, x|_S \models \theta\}$$

称为 S 中公式 θ 的含义。这样含义的自变量是语言的公式, 其值是系统中对象集合的子集。

下面这个命题解释了公式的含义。

命题 11.1

- (1) $|(a, v)|_S = \{x \in U: a(x)=v\}$;
- (2) $|\sim\theta|_S = -|\theta|_S$;
- (3) $|\theta \vee \psi|_S = |\theta|_S \cup |\psi|_S$;
- (4) $|\theta \wedge \psi|_S = |\theta|_S \cap |\psi|_S$;
- (5) $|\theta \rightarrow \psi|_S = -|\theta|_S \cup |\psi|_S$;
- (6) $|\theta \equiv \psi|_S = (|\theta|_S \cap |\psi|_S) \cup (-|\theta|_S \cup -|\psi|_S)$

可见公式的含义就是所有通过公式 θ 表达的对象集合, 或者说公式的含义是对象集 $|\theta|_S$ 的知识表达语言中的描述。

在我们的逻辑中, 我们也要用“真”这个概念, 当且仅当 $|\theta|_S = U$, 即系统中论域的所有对象满足该公式, 公式 θ 称为在知识表达系统 S 中为真。

当且仅当 $|\theta|_S = |\psi|_S$, 公式 θ 和 ψ 在 S 中是等价的。

下面的命题给出了所介绍概念的简单性质。

命题 11.2

- (1) 当且仅当 $|\theta|_S = U, | \models \theta$;

- (2) 当且仅当 $|\theta|_S = \emptyset, \models_S \sim \theta$;
- (3) 当且仅当 $|\theta|_S \subseteq |\Psi|_S, \models_S \theta \rightarrow \Psi$;
- (4) 当且仅当 $|\theta|_S = |\Psi|_S, \models_S \theta \equiv \Psi$ 。

最后，我们再次强调公式的含义依赖于我们对论域知识的了解，即依赖于知识表达系统，特别是一个公式在一个知识表达系统中可以为真，而在另一个知识表达系统中也可以为假，在我们的研究中，它们起了一个特殊的作用。

11.3.4 决策逻辑的推演

前面我们介绍的语言用来描述一个特殊知识表达系统所包含的知识，尽管对于许多知识表达系统，可以通过不同的对象集合作为共同的语言处理，但我们还是使用属性的同等集合与同等属性值集合来处理。从符号学的观点来看，所有这些系统的语言都是等同的，然而按照不同的对象的集合，它们的语义不同且它们在指定的知识表达系统中表示的性质也不同。

决策逻辑的所有公理集合由命题重复逻辑和一些特殊定理构成。在给出具体知识表达系统的公理之前，我们先引入一些辅助概念，使用下列缩写：

$$\begin{aligned}\theta \wedge \emptyset &=_{\text{df}} 0 \\ \theta \vee \emptyset &=_{\text{df}} 1\end{aligned}$$

显然， $\models 1$ 和 $\models \sim 0$ ，这样，1 和 0 可以分别代表真或假。

形式化公式通常定义如下：

$$(a_1, v_1) \wedge (a_2, v_2) \wedge \cdots \wedge (a_n, v_n)$$

这里， $v_i \in V_{a_i}, \{a_1, a_2, \cdots, a_n\} \in P$ ，且 $P \subseteq A$ ，上述形式公式称为 P 基本公式。对于属性常数集合 A，上述形式化公式就称为 A 基本公式。

当 $P \subseteq A$ ， θ 为 P 公式，且 $x \in U$ ，若 $x \models \theta$ ，则 θ 称为 S 中 x 的 P 描述，这与我们前面介绍的用条件属性集描述研究对象的出发点类似。

在知识表达系统 $S=(U, A)$ 中，所有可满足 A 基本公式的集合称为 S 中的基本知识。

用 $\Sigma(P)$ 表示 S 中的所有满足 P 公式的分解，当 $P=A$ 时，知识表达系统 $S=(U, A)$ 的特征公式定义为 $\Sigma(A)$ ，则 $\Sigma(A)$ 代表了 S 中的所有满足 A 基本公式的分解，表征了系统 S 中所包含的全部知识。具体地说，在我们的语言表中的每一行由一个特定的 A 基本公式表示，整个表由所有这样的公式的集合表示。

下面给出决策逻辑的一些公理：

- (1) 对于任一 $a \in A, v, u \in V_a$ ，且 $v \neq u$ ，存在 $(a, v) \wedge (a, u) = 0$;
- (2) 对于任一 $a \in A, v \in V_a$ ，存在 $v \vee (a, v) = 1$;
- (3) 对于任一 $a \in A, u \in V_a$ ，且 $v \neq u$ ，存在 $\sim(a, v) \equiv v \vee (a, u)$ 。

公理(1)是基于这样的假设：每一个对象对于每一个属性只有一个确切的值。例如某物为红色，它不可能是蓝色或绿色。

公理(2)是基于这样的假设：每一个属性对于系统中每一个对象必须有一个它的值域。例如当问题中的属性为颜色时，则每个对象必须有某种颜色，这种颜色就是这个属性的值。

公理(3)说明这样一个问题，即我们可以省略否定词，也就是我们可以说一个对象具有某一种性质，而不是说它不具有另外某些性质。例如我们可以说某物是绿色的或蓝色的，而不是说它不具有红色。

命题 11.3

对于任何 $P \subseteq A$,

$$|_S \Sigma_S(P) \equiv 1$$

命题 11.3 表明知识表达系统中所包含的知识是目前阶段所能获得的全部知识，且对应所谓的闭词假说。

当且仅当公式 θ 通过推理规则的有限运算，可以从公式 Ω 的公理和公式导出时，我们称公式 θ 是由一个公式 Ω 的集合可导出，记为 $\Omega \vdash \theta$ 。当公式 θ 由公理唯一可导出时，则公式 θ 称为决策逻辑的一个定理，简记为 $\vdash \theta$ 。

当且仅当公式 $\theta \wedge \sim \theta$ 是由 Ω 不可导时，公式 Ω 的集合称为一致的。

11.3.5 规范表达形式

知识表达语言中的公式可以用一种规范表达形式，这种形式与经典命题相似。

令 $P \subseteq A$ 为属性子集， θ 为知识表达语言中的一个公式。当且仅当 $\theta = \emptyset$ 或 $\theta = 1$ ，或者 θ 是一个 S 中非空 P 基本公式的一个分解（如果 $|\theta|_S \neq \emptyset$ ，公式 θ 是非空的），我们称 θ 是属于 S 中的 P 规范形式， A 规范形式则称为规范式。

命题 11.4

θ 为决策逻辑语言中的一个公式， P 包含 θ 所有的属性，若满足公理(1)~(3)和公式 $\Sigma_S(A)$ ，则 P 规范形式中有一个公式 ψ ，使得 $\vdash \theta \equiv \psi$ 成立。

可见，为计算公式的规范形式，我们可以通过命题推演和知识表达系统的一些公理来变换。

11.3.6 决策规则和决策算法

在逻辑语言中，含义 $\theta \rightarrow \Psi$ 称为知识表达语言中的决策规则， θ 和 Ψ 分别称为决策规则的前驱和后继，类似于前面我们所说明的用条件属性和决策属性描述研究对象，它们表达一种因果关系。

当 S 中决策规则 $\theta \rightarrow \Psi$ 为真时，我们说该决策规则是 S 中一致的，否则说该决策规则是 S 中不一致的。当决策规则是 S 中一致的时，相同的前驱必导致相同的后继；但同一种后继不一定必需是同一前驱产生的。

当 $\theta \rightarrow \Psi$ 为一个决策规则时，且 θ 和 Ψ 分别为 P 基本公式和 Q 基本公式， P, Q 已知，则决策规则 $\theta \rightarrow \Psi$ 称为 PQ 基本决策规则，简称 PQ 规则，这里的 P, Q 属性可以看成前面我们所说的条件属性和决策属性。

当 $\theta_1 \rightarrow \Psi, \theta_2 \rightarrow \Psi, \dots, \theta_n \rightarrow \Psi$ 均为基本决策规则时，决策规则 $\theta_1 \vee \theta_2 \vee \dots \vee \theta_n \rightarrow \Psi$ 称为基本决策规则 $\theta_1 \rightarrow \Psi, \theta_2 \rightarrow \Psi, \dots, \theta_n \rightarrow \Psi$ 的组合，简称为组合决策规则。

为了考察 PQ 规则是否为真（是否一致或不一致），我们可以利用下面的命题。

命题 11.5

当且仅当所有的 $\{P \vee Q\}$ 基本公式在 PQ 规则的前驱的 $\{P \vee Q\}$ 规范形式中出现，并且也在 PQ 规则的后继的 $\{P \vee Q\}$ 规范形式中出现，则 S 中的 PQ 规则为真（一致的），否则为假（不一致的）。

决策逻辑语言中任何有限决策规则集称为决策逻辑语言中的决策算法，而任何有限基本决策规则称为一个基本决策算法。

当基本决策算法中所有的决策规则都是 PQ 决策规则时，该算法称为 PQ 决策算法，或简称 PQ 算法，记作 (P, Q) 。当且仅当 S 中所有决策规则是一致的，则 S 中 PQ 算法是一致的，否则 PQ 算法是不一致的。

如果对于任一 $x \in U$ ，算法中存在一个 PQ 规则 $\theta \rightarrow \Psi$ ，使得 S 中 $x \models \theta \wedge \Psi$ ，则 S 中 PQ 算法是完备的，否则该算法是不完备的。

对于给定一个知识表达系统，系统中属性 P, Q 的非空子集可以唯一地确定一个 PQ 决策算

法，也就是说 PQ 算法和 PQ 决策表可以看作是等价的。

11.3.7 决策规则中的一致性和不分明性

为检验一个决策算法是否一致，我们必须考虑它的全部决策规则是否为真，我们可以利用命题 5，而下面的命题则给了一个更为简单的方法。

命题 11.6

当且仅当对于 PQ 决策算法中任一 PQ 决策规则 $\theta_1 \rightarrow \Psi_1$, $\theta = \theta_1$, 蕴含 $\Psi = \Psi_1$, 则 PQ 决策算法中 PQ 决策规则 $\theta \rightarrow \Psi$ 是 S 中一致的。

同时我们注意到，为了检验决策规则 $\theta \rightarrow \Psi$ 是否为真，我们必须证明该决策规则的前驱（公式 θ ）能够将决策类 Ψ 同所讨论的决策算法的其他决策类区分开，这表明“真”这一概念有时也可以用不分明性的概念代替。显然，若相同的前驱有不同的后继，则这种规则是不一致的。

11.4 决策表的约简

在我们制定决策时是否需要全部的条件属性，能否进行决策表的约简。约简后的决策表具有与约简前的决策表相同的功能，但是约简后的决策表具有更少的条件属性。因此决策表的约简在工程应用中相当重要，同样的决策可以通过基于更少量的条件，使我们通过一些简单的手段就能获得同样要求的结果。严格地说，虽然决策算法和决策表是两个不同的概念，但用决策表表示决策算法比用决策逻辑语言的形式更紧凑，更易于理解，是一种简便的方法，同样，决策算法也可以从逻辑方面来表达决策表，所以两者的有些性质可以相互利用。

11.4.1 属性的依赖性

要处理数据，进行决策，就要分析数据的内在联系，讨论属性的依赖性。这里所讨论的属性的依赖性和我们前面介绍的知识的依赖性是对应的。当 S 中存在一一致的 PQ 决策算法时，我们称 S 中属性集 Q 全依赖于（简称依赖）属性集 P，并记作 $P \Rightarrow Q$ ；当 S 中存在一不一致的 PQ 决策算法时，我们称 S 中属性集 Q 部分依赖于属性集 P。

如前面的知识的依赖性的定义，我们也可以利用正域的概念来定义属性集之间的依赖度。当 (P, Q) 为 S 中的一个 PQ 算法，算法中所有一致的 PQ 规则的集合称为算法的正域，记为 $POS(P, Q)$ 。决策算法的正域 $POS(P, Q)$ 是不一致算法的一致部分，显然，当且仅当 $POS(P, Q) \neq (P, Q)$ 或 $card(POS(P, Q)) \neq card((P, Q))$ 时，算法是不一致的。

对于一个 PQ 算法，算法的一致性程度用依赖度 k 来表示，并定义为：

$$k = card(POS(P, Q)) / card(P, Q)$$

显然， $0 \leq k \leq 1$ 。当 $k=1$ 时，算法是一致的；当 $k \neq 1$ 时，算法是不一致的。当 PQ 算法有依赖度 k 时，我们称属性 Q 对 P 的依赖度为 k，并记为 $P \Rightarrow_k Q$ 。

11.4.2 一致决策表的约简

我们知道，利用不分明性可以研究知识的约简，即在 S 中存在属性集 $C \subseteq A$ ，当且仅当 $IND(A-C) = IND(A)$ 时 A 是冗余的，当 A-C 在 A 中冗余且 C 在 S 中是依赖的时，则 C 是 A 的约简。

在这里，我们讨论的问题将用逻辑方式表示，并且利用算法的一致性来做出判断和进行约简。

(P, Q) 为一致算法， $a \in P$ ，在 (P, Q) 算法中，当且仅当 $((P - \{a\}), Q)$ 算法为一致的时，我们称 (P, Q) 算法中属性 a 是可省的，否则 a 为不可省的。

如果所有的属性 $a \in P$ 是 (P, Q) 算法中不可省的, 则算法 (P, Q) 称为独立的。如果属性子集 $R \subseteq P$, 当算法 (R, Q) 是独立且一致的, 则 R 称为算法 (P, Q) 中 P 的约简。如果属性子集 R 为算法 (P, Q) 中 P 的约简时, 算法 (R, Q) 称为算法 (P, Q) 的约简, 算法的约简是去掉不必要的条件属性, 是对知识表示空间维数进行约简。

算法 (P, Q) 中所有不可省的属性的集合称为算法 (P, Q) 的核, 记作 $CORE(P, Q)$ 。

命题 11.7

$$CORE(P, Q) = \cap RED(P, Q)$$

这里, $RED(P, Q)$ 代表算法 (P, Q) 所有约简的集合。

另外, 我们再介绍一下决策表中属性的一些性质:

命题 11.8

当且仅当 $C \Rightarrow D$, 决策表 $T=(U, A, C, D)$ 是一致的。

由命题 1, 很容易通过计算条件属性和决策属性间的依赖程度来检查一致性。当依赖程度等于 1 时, 我们说决策表是一致的, 否则不一致。

命题 11.9

每个决策表 $T=(U, A, C, D)$ 都可以唯一分解为两个决策表 $T_1=(U_1, A, C, D)$ 和 $T_2=(U_2, A, C, D)$, 这样使得表 T_1 中 $C \Rightarrow_1 D$ 和 T_2 中 $C \Rightarrow_0 D$ 。这里 $U_1 = POS_C(D)$, $U_2 = \cup BN_C(X)$, $X \in U \setminus IND(D)$ 。

由命题 2 可见, 假设我们已计算出条件属性的依赖度, 若表的结果不一致, 即依赖度小于 1, 则由命题 2 可以将表分解成两个子表: 其中一个表完全不一致, 依赖度为 0; 另一个表则完全一致, 依赖度为 1。当然, 只有依赖度大于 0 且不等于 1 时, 这一分解才能进行。

决策表的约简步骤如下:

- (1) 对决策表进行条件属性的约简, 即从决策表中消去某一列;
- (2) 消去重复的行;
- (3) 消去每一决策规则中属性的冗余值。

应该注意到, 与知识表达系统的一般表达相比, 这里的行不表示对任何实际对象的描述, 因此重复行表示的是同样的决策, 所以可以把它消去。

约简后的决策表是一个不完全的决策表, 它仅包含那些在决策时所必需的条件属性值, 但它具有原始知识系统的所有知识。

11.4.2.1 条件属性的约简

A.Skowron 提出了用分明矩阵(Discernibility Matrix)的形式来表示知识的方法, 这种表示方法有很多优点, 尤其使核与约简等概念的计算较为简单, 其主要思想为:

设 $S=(U, A)$ 为一个知识表示系统, 其中 $U = \{x_1, x_2, \dots, x_n\}$, x_i 为所讨论的个体, $i=1, 2, \dots, n$, $A = \{a_1, a_2, \dots, a_m\}$, a_j 为个体所具有的属性, $j=1, 2, \dots, m$ 。

知识表达系统 S 的分明矩阵 $M(S)=[c_{ij}]_{n \times n}$, 其中矩阵项定义如下:

$$c_{ij} = \{a \in A: a(x_i) \neq a(x_j), i, j=1, 2, \dots, n\}$$

因此 c_{ij} 是个体 x_i 与 x_j 有区别的所有属性的集合, 利用分明矩阵可以方便地求解属性集合 A 的核与约简。

核就可以定义为分明矩阵中所有只有一个元素的矩阵项的集合, 即

$$CORE(A) = \{a \in A: c_{ij} = (a), \text{ 对一些 } i, j\}$$

我们也很容易看出相对于集合包含关系运算而言, 若属性集合 $B \subseteq A$ 是满足下列条件

$$B \cap c_{ij} \neq \emptyset, \text{ 对于 } M(S) \text{ 中的任一非空项 } c_{ij} \neq \emptyset$$

的一个最小属性子集, 则称属性集合 $B \subseteq A$ 是 A 的一个约简。换言之, 约简是这样的最小属性子集, 它能够区分用整个属性集合 A 可区分的所有对象。

由于 $M(S)$ 是对称的, 且对于每个 $i=1, 2, \dots, n$, $c_{ii} = \emptyset$, 所以可以用 $M(S)$ 的下三角来表示 $M(S)$,

$$1 \leq j < i \leq n.$$

对于每一个分明矩阵 $M(S)$ 对应唯一的分明函数 $f_{M(S)}$ (Discernibility Function), 它的定义如下:

信息系统 S 的分明函数 $f_{M(S)}$ 是一个有 m -元变量 $a_1, \dots, a_m (a_i \in A, i=1, \dots, m)$ 的布尔函数, 它是 $\bigvee c_{ij}$ 的合取, $\bigvee c_{ij}$ 是矩阵项 c_{ij} 中的各元素的析取, $1 \leq j < i \leq n$ 且 $c_{ij} \neq \Phi$ 。

根据分明函数与约简的对应关系, A.Skowron 提出了计算信息系统 S 的约简 $RED(S)$ 的方法:

- 1) 计算信息系统 S 的分明矩阵 $M(S)$
- 2) 计算与分明矩阵 $M(S)$ 对应的分明函数 $f_{M(S)}$
- 3) 计算分明函数 $f_{M(S)}$ 的最小析取范式, 其中每个析取分量对应一个约简

需要说明的是该方法可以求出所有的约简, 但是只适合于非常小的数据集。

为了对决策表进行约简, 我们可以采用分明矩阵的方法对条件属性进行约简, 只不过我们是通过决策属性来得到等价类, 对决策属性相同的个体不予比较。考虑决策表 11.1。

表 11.1 信息表

$U \setminus A$	a	b	c	d	e
u_1	1	0	2	1	0
u_2	0	0	1	2	1
u_3	2	0	2	1	0
u_4	0	0	2	2	2
u_5	1	1	2	1	0

其中条件属性为 a, b, c, d , 决策属性 e , 则其对应的分明矩阵为

表 11.2 分明矩阵

u	u_1	u_2	u_3	u_4	u_5
u_1					
u_2	a, c, d				
u_3		a, c, d			
u_4	a, d	c	a, d		
u_5		a, b, c, d		a, b, d	

由此分明矩阵很容易得到核为 $\{c\}$, 分明函数 $f_{M(S)}$ 为 $c \wedge (a \vee d)$, 即 $(a \wedge c) \vee (c \wedge d)$, 得到两个约简 $\{a, c\}$ 和 $\{c, d\}$ 。

当讨论的对象与属性的规模较大时, 矩阵将占有大量的存储空间。但是经过分析, 分明矩阵起到解释的作用, 比较直观, 其实质是利用逻辑运算中的吸收律和其他演算法则来达到数据约简的目的, 所以可以省略生成分明矩阵的中间环节, 从表中提取关于属性的逻辑公式, 采用基于广义决策逻辑公式演绎算法, 也就是利用决策表直接提取逻辑公式, 并在逻辑演绎系统下化简该公式, 即一边提取那些关于个体是分明的属性 (决策属性相同的个体不进行比较) 并生成公式, 一边化简该公式, 最后将公式化为析取范式, 则每个析取项对应一个约简, 达到属性约简的目的。

u_1 与 u_2, u_3, u_4, u_5 关于属性的值是分明的分明合取范式: $(a \vee c \vee d) \wedge (a \vee d)$

经引用吸收律等逻辑运算得到简化式①: $a \vee d$

u_2 与 u_3, u_4, u_5 关于属性的值是分明的分明合取范式并加上①:

$$(a \vee d) \wedge (a \vee c \vee d) \wedge c \wedge (a \vee b \vee c \vee d)$$

经引用吸收律等逻辑运算得到简化式②: $(a \vee d) \wedge c$

u_3 与 u_4, u_5 关于属性的值是分明的分明合取范式并加上②:

$$(a \vee d) \wedge c \wedge (a \vee d)$$

经引用吸收律等逻辑运算得到简化式③: $(a \vee d) \wedge c$

u_4 与 u_5 关于属性的值是分明的分明合取范式并加上③:

$$(a \vee d) \wedge c \wedge (a \vee b \vee d)$$

经引用吸收律等逻辑运算得到简化式④: $(a \vee d) \wedge c$

对④式施行 \wedge 对 \vee 的分配律运算, 得到如下的分明析取范式: $(a \wedge c) \vee (c \wedge d)$

由此得到两个约简: $\{a, c\}$ 和 $\{c, d\}$ 。

这样, 表 11.1 就可以简化为如下的两个表:

表 11.3

表 11.1 对应的一种简化决策表			
$U \setminus A$	a	c	e
u_1	1	2	0
u_2	0	1	1
u_3	2	2	0
u_4	0	2	2
u_5	1	2	0

表 11.4

表 11.1 对应的一种简化决策表

$U \setminus A$	c	d	e
U_1	2	1	0
U_2	1	2	1
U_3	2	1	0
U_4	2	2	2
U_5	2	1	0

11.4.2.2 行的约简

对决策表中的重复的行要删除, 因为它们的条件属性和决策属性都相同, 都表示同一条决策规则。另外, 决策规则的列表顺序不是本质性的, 所以表 11.3、表 11.4 都可进行的约简, 如表 11.3 可化为:

表 11.5 表 11.3 的化简

$U \setminus A$	a	c	e
u_1	1	2	0
u_2	0	1	1
u_3	2	2	0
u_4	0	2	2

11.4.2.3 属性值的约简

对于决策表而言, 属性值的约简就是决策规则的约简。决策规则的约简是利用决策逻辑分别通过消去决策算法中每个决策规则的不必要条件, 它不是整体上约简属性, 而是针对每个决策规则, 去掉表达该规则时的冗余属性值, 以便进一步使决策算法最小化。

前面我们已经定义, 当 θ 为 P 基本公式且 $R \subseteq P$ 时, 那么 θR 表示通过从 θ 中消去全部初等公式 (a, V_a) 得到的 R 基本公式, 且 $a \in P - R$ 。

$\theta \rightarrow \Psi$ 为 PQ 规则, 且 $a \in P$, 当且仅当 $\models_S \theta \rightarrow \Psi$ 蕴含 $\models_S \theta(P - \{a\}) \rightarrow \Psi$, 我们称属性 a 是规则 $\theta \rightarrow \Psi$ 中可省略的。否则, 属性 a 是规则 $\theta \rightarrow \Psi$ 中不可省略的。

在规则 $\theta \rightarrow \Psi$ 中, 若所有属性 a 是规则 $\theta \rightarrow \Psi$ 中不可省略的, 则 $\theta \rightarrow \Psi$ 是独立的。当 $\theta \rightarrow \Psi$ 是独立的, 且 $\models_S \theta \rightarrow \Psi$ 蕴含 $\models_S \theta R \rightarrow \Psi$, 属性子集 $R \subseteq P$ 称为 PQ 规则 $\theta \rightarrow \Psi$ 的约简。当 R 是 PQ 规则 $\theta \rightarrow \Psi$ 的约简时, $\theta R \rightarrow \Psi$ 叫做被约简。

$\theta \rightarrow \Psi$ 中所有可省略的属性集称为规则 $\theta \rightarrow \Psi$ 的核, 记作 $CORE(\theta \rightarrow \Psi)$

命题 11.10 $CORE(P \rightarrow Q) = \bigcap RED(P \rightarrow Q)$

这里 $RED(P \rightarrow Q)$ 是所有 $P \rightarrow Q$ 约简的集合。

正如我们所说, 决策规则的约简就是利用决策逻辑消去决策算法中每个决策规则的不必要条件, 即计算算法中的每个决策规则的核和约简。

对决策表进行属性值的约简, 也就是决策规则的约简, 实际上是针对条件属性而言。而每一行都对应一条决策规则, 所以要计算某决策规则的条件属性的核值, 可先把一个该行中条件属性的值从表中删去, 然后看剩下的该行中条件属性的值是否唯一决定此行中的决策属性, 若不是, 这个值就是核值; 求完某决策规则的条件属性值的核值后, 再去求它的条件属性值的约简, 即将一些条件属性值加入到核值中, 得到的各条件属性值能保证表的一致性, 并且每个条件属性值都不可省。若在决策表的约简表中也出现了重复行, 也应该将其删除, 因为它们还是表示相同的决策规则。

以表 11.5 为例, 在第一条决策规则 $a_1 c_2 \rightarrow e_0$ 中, 其中 a_1 是核值, 因为规则 $a_1 \rightarrow e_0$ (去掉 c_2) 为真, 而 $c_2 \rightarrow e_0$ (去掉 a_1) 为假, 故不能去掉 a_1 。按此种方法计算, 表 5 中每一决策规则的所有核值如表 11.6 所示。

表 11.6 表 11.5 的核值表

$U \setminus A$	a	c	e
u_1	1	—	0
u_2	—	1	1
u_3	2	—	0
u_4	0	2	2

求完所有的核值后, 再求每一决策规则的约简。如第一条决策规则有一个约简: $a_1 \rightarrow e_0$, 因为这个决策规则能保持表的一致性。如此可求得表 11.5 的约简表。

表 11.7 表 11.5 的约简表

$U \setminus A$	a	c	e
u_1	1	×	0
u_2	×	1	1
u_3	2	×	0
u_4	0	2	2

该表对应的决策算法为：

$$a_1 \vee a_2 \rightarrow e_0, \quad c_1 \rightarrow e_1, \quad a_0 c_2 \rightarrow e_2,$$

11.4.2.4 属性的重要性

前面已经提到，约简是粗糙集用于数据分析的重要概念，而所有约简的计算是 NP-hard 问题，因此运用启发信息来简化计算以找出最优或次优约简是必要的。

现在在求约简的算法一般都使用核作为计算约简的出发点，计算一个最好的或者用户指定的最小约简。算法将属性的重要性作为启发规则，按照属性的重要度从大到小逐个加入属性，直到该集合是一个约简为止。各算法对属性重要度的度量不同，目前的报道涉及到以下几种度量：

(1) 根据依赖度的变化来定义

设 S 为一决策表， C 、 D 分别为条件属性集或决策属性集。 $R \subset C$ ，对于任意属性 $a \in C - R$ 的重要性定义如下：

$$SGF(a, R, D) = k(R \cup \{a\}, D) - k(R, D)$$

其中， $k(R, D) = \text{card}(\text{POS}_R(D)) / \text{card}(\text{POS}_C(D))$ 。

还有一种定义为：

$$SGF(a, R, D) = \gamma_{R \cup \{a\}} - \gamma_R$$

其中， $\gamma_R = \text{card}(\text{POS}_R(D)) / \text{card}(U)$ 。

以上两种定义实质是相同的，因为当决策表给定后， $\text{card}(\text{POS}_C(D))$ 和 $\text{card}(U)$ 都是常数，尽管重要性的取值不同，但是它们对属性重要性的排序是相同的。

(2) 根据信息熵来定义

假设 $H(D/R)$ 为 D 相对于 R 的条件熵，属性 a 的重要性定义如下：

$$SGF(a, R, D) = H(D/R) - H(D/R \cup \{a\})$$

(3) 根据在分明矩阵中出现的频率

设 M 是根据决策表 S 构造的分明矩阵，令 $p(a)$ 为在 M 中属性 a 的属性频率函数，它定义为 a 在 M 中出现的次数，则

$$SGF(a, R, D) = p(a)$$

11.4.3 非一致决策表的约简

只有当所有的决策规则都是一致时，决策表才是一致的，否则决策表是不一致的。对于一致的决策表比较容易处理，在进行约简时，只要判断去掉某个属性或某个属性值时是否会导致不一致规则的产生。而对不一致表进行约简时就不能再使用这种方法了，本章将介绍两种化简不一致表的方法：一种是考虑正域的变化，另外一种是将不一致表分成完全一致表和完全不一致表两个子表。

非一致决策表的约简步骤与一致决策表的约简步骤类似，值得注意的是，前面提到一致决策表的约简步骤，其中有关于重复行的删除，即相同的决策规则可以删除。但在不一致决策表中，这种处理方法将要视决策规则的一致性而定，若决策规则是一致的，则可删除；若是非一致的，则不能删除，下面将给以具体说明。

11.4.3.1 考虑正域的变化

对于知识表达中不一致的情况，也可以类似一致的情况那样进行决策表的约简，即考虑去掉某些属性后其正域是否发生变化，以判断该属性是否可以去掉。若 (P, Q) 为不一致算法， $a \in P$ ，当 $\text{POS}(P, Q) = \text{POS}(P - \{a\}, Q)$ 时， (P, Q) 算法中属性 a 是可省的，否则属性 a 在 (P, Q) 中是

不可省的。

当 (P, Q) 算法中所有的属性 $a \in P$ 是不可省的,则算法 (P, Q) 称为独立的。如果属性子集 $R \subseteq P$,当算法 (R, Q) 是独立的且 $POS(P, Q) = POS(R, Q)$ 时,属性子集 R 称为算法 (P, Q) 中 P 的约简。

算法 (P, Q) 中所有不可省的属性的集合称为算法 (P, Q) 的核,记作 $CORE(P, Q)$ 。

可见一致算法是不一致算法的特殊情况。

而对决策规则进行约简时,也就是进行条件属性值的约简时,则要分别处理:

若该决策规则是一致的,首先要计算它的条件属性的核值,即把该行中一个条件属性的值从表中删去,然后看该行中剩下的条件属性的值是否唯一决定此行中的决策属性,若不是,这个值就是核值;求完其条件属性值的核值后,再去求其条件属性值的约简,即将一些条件属性值加入到核值中,得到的各条件属性值能保证决策规则的一致性,并且每个条件属性值都不可省。

若该决策规则是非一致的,首先要计算它的条件属性的核值,即把该行中一个条件属性的值从表中删去,然后看根据该行中剩下的条件属性的值所得到的决策属性值的集合与未删除前所得到的决策属性值的集合是否相同,若不是,这个值就是核值;求完其条件属性值的核值后,再去求其条件属性值的约简,即将一些条件属性值加入到核值中,保证得到的决策属性值的集合与原来的决策属性值的集合相同,并且每个条件属性值都不可省。

下面我们举例说明采用这种方法如何对不一致决策表进行约简。

考虑下面的一个知识表达系统:

表 11.8 一个知识表达系统的决策表

U	a	b	c	d	e
1	1	0	2	2	0
2	0	1	1	1	2
3	2	0	0	1	1
4	1	1	0	2	2
5	1	0	2	0	1
6	2	2	0	1	1
7	2	1	1	1	2
8	0	1	1	0	1

这里, $C=\{a, b, c\}$ 和 $D=\{d, e\}$ 分别为条件属性和决策属性,因为 $U|C=\{\{1,5\}, \{2,8\}, \{3\}, \{4\}, \{6\}, \{7\}\}$, $U|D=\{\{1\}, \{2,7\}, \{3,6\}, \{4\}, \{5,8\}\}$, $POS_C(D)=\{\{3\}, \{4\}, \{6\}, \{7\}\}$, $r_C(D)=4/8 \neq 1$,这表明表 11.8 是不一致的。

去掉条件属性 a 后, $U|(C-\{a\})=\{\{1,5\}, \{2,7,8\}, \{3\}, \{4\}, \{6\}\}$, $POS_{C-\{a\}}(D)=\{\{3\}, \{4\}, \{6\}\} \neq POS_C(D)$,所以属性 a 是不可省的;去掉条件属性 b 后, $U|(C-\{b\})=\{\{1,5\}, \{2,8\}, \{3,6\}, \{4\}, \{7\}\}$, $POS_{C-\{b\}}(D)=\{\{3\}, \{4\}, \{6\}, \{7\}\} = POS_C(D)$,所以属性 b 是可省的;去掉条件属性 c 后, $U|(C-\{c\})=\{\{1,5\}, \{2,8\}, \{3\}, \{4\}, \{6\}, \{7\}\}$, $POS_{C-\{c\}}(D)=\{\{3\}, \{4\}, \{6\}, \{7\}\} = POS_C(D)$,所以属性 c 是可省的。

由此可得到,表 11.8 的条件属性的核为 a ,有两个约简 $\{a, b\}$ 和 $\{a, c\}$ 。考虑约简 $\{a, b\}$ 对应的表 11.9。

表 11.9 去掉条件属性 b 后的决策表

U	a	d	e
-----	-----	-----	-----

1	1	0	2	0
2	0	1	1	2
3	2	0	1	1
4	1	1	2	2
5	1	0	0	1
6	2	2	1	1
7	2	1	1	2
8	0	1	0	1

对表 11.9 进行属性值的约简，对于第三条决策规则 $a_2b_0 \rightarrow d_1e_1$ ，它为一致决策规则，其中 a_2, b_0 是核值，因为规则 $a_2 \rightarrow d_1e_1$ （去掉 b_0 ）和 $b_0 \rightarrow d_1e_1$ （去掉 a_2 ）都为非一致规则。

对于第二条决策规则 $a_0b_1 \rightarrow d_1e_2$ ，它为一致决策规则，其中 a_0 是核值，因为 a_0b_1 对应的决策属性集为 $\{d_1e_2, d_0e_1\}$ ，去掉 b_1 后， a_0 对应的决策属性集还是 $\{d_1e_2, d_0e_1\}$ ；而去掉 a_0 后， b_1 对应的决策属性集是 $\{d_1e_2, d_2e_2, d_0e_1\}$ ， a_0 故不能去掉。按此种方法计算，表 11.9 中每一决策规则的所有核值如表 11.10 所示。

表 11.10 表 11.9 条件属性的核值表

U	a	b	d	e
1	1	0	2	0
2	0	—	1	2
3	2	0	1	1
4	1	1	2	2
5	1	0	0	1
6	—	2	1	1
7	2	1	1	2
8	0	—	0	1

因此，我们得到所有条件属性值的约简如表 11.11 所示：

表 11.11 表 11.9 条件属性的值约简表

U	A	b	d	e
1	1	0	2	0
2	0	×	1	2
3	2	0	1	1
4	1	1	2	2
5	1	0	0	1
6	×	2	1	1
7	2	1	1	2
8	0	×	0	1

设 θ 和 Ψ 分别表示条件和决策的逻辑公式， $\theta \rightarrow \Psi$ 是决策规则。我们用 $| \Psi |$ 表示在 S 中满足公式的个体的集合，而且我们可以在每一条决策规则中带一个数值，称此为规则的粗糙算子，它被定义为： $\mu(\theta, \Psi) = K(|\theta \wedge \Psi|) / K(|\theta|)$ ，带粗糙算子的规则的形式为： $\theta \rightarrow_m \Psi$ ， $m = \mu(\theta, \Psi)$ 。

其中， $K(S)$ 表示集合 S 的基数，与数学中用 $|S|$ 表示有相同意义，即集合 S 中的元素个数。显然， $0 \leq \mu(\theta, \Psi) \leq 1$ 。如果这条规则 $\theta \rightarrow \Psi$ 是完全一致的，粗糙算子则可省去。

因此，表 11.11 对应的决策算法为：

$$\begin{aligned}
&a_1b_0 \rightarrow_{0.5} d_2e_0, \\
&a_0 \rightarrow_{0.5} d_1e_2, \\
&a_2b_1 \rightarrow d_1e_2, \\
&a_2b_0 \vee b_2 \rightarrow d_1e_1, \\
&a_0 \rightarrow_{0.5} d_0e_1, \\
&a_1b_1 \rightarrow d_2e_2, \\
&a_1b_0 \rightarrow_{0.5} d_0e_1.
\end{aligned}$$

11.4.3.2 分成两个子表

每个决策表 $T=(U, A, C, D)$ 都可以唯一分解为两个决策表 $T_1=(U_1, A, C, D)$ 和 $T_2=(U_2, A, C, D)$, 这样使得表 T_1 中 $C \Rightarrow_1 D$ 和 T_2 中 $C \Rightarrow_0 D$ 。这里 $U_1=POS_C(D)$, $U_2=\cup BN_C(X)$, $X \in U \setminus IND(D)$ 。

由此命题可见, 假设我们已计算出条件属性的依赖度, 若表的结果不一致, 即依赖度小于 1, 则可以将表分解成两个子表: 其中一个表完全不一致, 依赖度为 0; 另一个表则完全一致, 依赖度为 1。当然, 只有依赖度大于 0 且不等于 1 时, 这一分解才能进行。分解后所得到的完全一致表可按照第四章所提到的方法进行约简, 而对完全不一致表可不处理, 直接生成带粗糙算子的决策规则。

再依次考虑上面的例 11.1, 表 11.1 可分解为如下的两个子表:

表 11.12 完全一致的决策表

U	a	b	c	d	e
3	2	0	0	1	1
4	1	1	0	2	2
6	2	2	0	1	1
7	2	1	1	1	2

表 11.13 完全不一致的决策表

U	a	b	c	d	e
1	1	0	2	2	0
2	0	1	1	1	2
5	1	0	2	0	1
8	0	1	1	0	1

对于表 11.12, $U \setminus C = \{\{3\}, \{4\}, \{6\}, \{7\}\}$, $U \setminus D = \{\{3,6\}, \{4\}, \{7\}\}$, $POS_C(D) = \{\{3\}, \{4\}, \{6\}, \{7\}\}$, $r_C(D) = 4/4 = 1$, 因此表 11.12 是完全一致的, 该表中所有的决策规则是一致的。

对于表 11.13, $U \setminus C = \{\{1, 5\}, \{2, 8\}\}$, $U \setminus D = \{\{1\}, \{2\}, \{5, 8\}\}$, $r_C(D) = 0/4 = 0$, 因此表 11.6 是完全不一致的, 该表中对应的决策规则都是不一致的。

对表 11.12, 我们采用第四章所提的方法进行约简, 得到它对应的条件属性的约简为 $\{a, b\}$ 、 $\{a, c\}$ 和 $\{b, c\}$ 。取约简 $\{a, b\}$, 对应的表为:

表 11.14 表 11.11 去掉条件属性 c 的决策表

U	a	b	d	e
3	2	0	1	1
4	1	1	2	2
6	2	2	1	1

7	2	1	1	2
---	---	---	---	---

对表 11.13 进行条件属性值的约简，得到其对应的决策算法：

$$b_0 \vee b_2 \rightarrow d_1 e_1,$$

$$a_1 \rightarrow d_2 e_2,$$

$$a_2 b_1 \rightarrow d_1 e_2。$$

对表 11.12 不进行处理，直接生成带粗糙算子的决策规则：

$$a_1 b_0 c_2 \rightarrow_{0.5} d_2 e_0,$$

$$a_0 b_1 c_1 \rightarrow_{0.5} d_1 e_2,$$

$$a_1 b_0 c_2 \rightarrow_{0.5} d_0 e_1,$$

$$a_0 b_1 c_1 \rightarrow_{0.5} d_0 e_1,$$

最后，将所有的对应完全一致决策表的决策算法与对应于完全不一致决策表的决策算法合并，就得到了对应于原来的非一致决策表的决策算法。

11.5 粗糙集的扩展模型

基本粗糙集理论和其他处理不精确与不确定性的方法相比具有独特之处，然而仍然存在着某些片面性与不足之处。目前，大多数成功的应用，都从不同的侧面对基本粗糙集理论进行了拓广。基本粗糙集理论是假设对于已知的对象全域拥有必要知识的前提之下的，是处理模糊性和不确定性的一种数学工具，本质上可认为是一种三值逻辑（正区域、边界区域和负区域）。

基本粗糙集理论的主要存在的问题是：

- 1) 对原始数据本身的模糊性缺乏相应的处理能力；
- 2) 对于粗糙集的边界区域的刻画过于简单；
- 3) 粗糙集理论的方法在可用信息不完全的情况下将对象们归类于某一具体的

类，通常分类是确定的，但并未提供数理统计中所常用的在一个给定错误率的条件下将尽可能多的对象进行分类的方法，而实际中常常遇到这类问题。

在粗糙集理论应用研究中已提出了许多扩展模型，例如可变精度粗糙集模型(Variable Precision 粗糙 Set—VPRS)模型，一些基于粗糙集的非单调逻辑模型，以及与粗糙集理论或证据理论相结合的模型等。下面，着重评述几个典型的拓广模型。

11.5.1 可变精度粗糙集模型

原始粗糙集模型假设全域 U 是已知的，所推出的结论仅适用于 U 中的对象。实际应用中，满足此条件是非常困难的。为解决这个矛盾，就必须寻求一种方法，能够从少量样本中得出结论，并应用到更大的范围中，而且此结论仅在样本范围内是正确的，在扩展到的论域上，则应被看作不确定的或模糊的。为此，W.Ziarko 提出了一种称之为可变精度粗糙集模型，该模型给出了错误率低于预先给定值的分类策略，定义了该精度下的正区域、边界区域和负区域，讨论了此定义下的有关性质。J.D.Katzberg 和 W.Ziarko 进一步提出了不对称边界的 VPRS 模型，使此模型更加一般化，从而拓广了 VPRS 的应用范围。下面扼要介绍 VPRS 的思想。

一般地，集合 X 包含于 Y 并未反映出集合 X 的元素属于集合 Y 的“多少”。为此，VPRS 定义了它的量度：

$$C(X, Y) = 1 - \text{card}(X \cap Y) / \text{card}(X) \quad \text{当 } \text{card}(x) > 0,$$

$$C(X, Y) = 0 \quad \text{当 } \text{card}(x) = 0。$$

$C(X, Y)$ 表示把集合 X 归类于集合 Y 的误分类度, 即有 $C(X, Y) \times 100\%$ 的元素归类错误。显然, $C(X, Y)=0$ 时有 $X \subseteq Y$ 。如此, 可事先给定一错误分类率 $\beta (0 \leq \beta < 0.5)$, 基于上述定义, 我们有 $X \subseteq^\beta Y$, 当且仅当 $C(X, Y) \leq \beta$ 。

在此基础上, 设 U 为论域且 R 为 U 上的等价关系, $U/R=A=\{X_1, X_2, \dots, X_k\}$, 这样, 可定义集合 X 的 β -下近似为

$$R_\beta X = \cup X_i \quad (X_i \subseteq^\beta X, i=1, 2, \dots, k)$$

或

$$R_\beta X = \cup X_i \quad (C(X_i, X) \leq \beta, i=1, 2, \dots, k),$$

并且 $R_\beta X$ 称为集合 X 的 β -正区域, 集合 X 的 β -上近似为

$$R^\beta X = \cup X_i \quad (C(X_i, X) < 1-\beta, i=1, 2, \dots, k),$$

这样, β -边界区域就定义为

$$BNR_\beta X = \cup X_i \quad (\beta < C(X_i, X) < 1-\beta);$$

β -负区域为

$$NEGR_\beta X = \cup X_i \quad (C(X_i, X) \geq 1-\beta)。$$

以此类推, 我们还可以定义 β -依赖、 β -约简等与传统粗糙集模型相对应的概念。

可以看出, 当 $\beta=0$ 时, $VPRS$ 模型就蜕化为传统的粗糙集模型, 即粗糙集为 $VPRS$ 的一个特例。另外值得一提的是, $VPRS$ 是粗糙集的直接扩展, 它完全继承了粗糙集的性质, 拥有粗糙集的所有优点并拓展了粗糙集理论的应用范围。

11.5.2 相似模型

在数据中存在缺失的属性值的时候 (在数据库中很普遍), 不分明关系或等价关系无法处理这种情形。为扩展粗糙集的能力, 有许多作者提出了用相似关系来代替不分明关系作为粗糙集的基础。

在使用相似关系代替粗糙集的不分明关系后, 最重要的变化就是相似类不再形成对集合的划分了, 它们之间是相互重叠的。类似于等价类, 可以定义相似集, 即所有和某各元素 x 在属性集合 B 上相似的集合 $SIM_b(x)$ 。值得注意的是 $SIM_b(x)$ 中的元素不一定属于同一决策类, 因此还需要定义相似决策类, 即相似集对应的决策类集合。

由于相似集的元素并不一定属于同一决策类, 为此定义相对吸收集。子集 $Y \subseteq U$ 称为相对吸收集, 如果对于每个 $x \in U$, 存在 $y \in Y$ 与之相似, 并且具有同样的决策值。显然相对吸收集可以用来进行数据约简。利用相似集可以很容易地定义正区域地概念, 它就是所有包含在决策类中的相似集的并, 依赖度和约简的概念都可以类似经典集合的方式定义。

实践证明, 相似模型在实践使用中具有比经典粗糙集模型更好的性能。在解决数据库中缺少值的情况时, 一个简单的相似关系可以定义为 (其中 ? 代表不知道或不关心):

$$\tau_c(x, y) = \{x \in U, y \in U | \forall a \in C, a(x) = a(y) \text{ or } a(x) = ? \text{ or } a(y) = ?\}$$

11.5.3 基于粗糙集的非单调逻辑

自粗糙集理论提出以来, 粗糙集理论的研究者都很重视它的逻辑研究, 试图通过粗糙集建立粗糙逻辑, 也相应地发表了一系列的粗糙逻辑方面的论文。如 Z.Pawlak 于 1987 年发表了题为“粗糙 Logic”的论文, 他在这篇论文中给出了其逻辑公式的语义解释: 真、假、粗糙真、粗糙假和粗糙非一致性。这 5 种值可视为不同的近似程度, 但它们缺乏确切的数学描述。Z.Pawlak 等人在 1995 年第 11 期 *Communication of The ACMS* 上的综述中认为研究粗糙逻辑——基于粗糙集的不精确推理逻辑——可能是最重要的课题。

T.Y.Lin 和 Q.Liu 等基于拓扑学观念定义了粗糙下近似算子 L 和粗糙上近似算子 H , 这两

个算子的语法性质分别与模态逻辑的必然算子 \Box 和可能算子 \Diamond 十分相似,因而带有 L 和 H 算子的逻辑公式被称为粗糙逻辑公式,并且建立了与模态逻辑相似的公理化粗糙逻辑演绎系统和相平行的演绎规则,但由于其定义的一阶粗糙逻辑在语义上,就 L 和 H 而言是含糊的,无法从数学上给出解释。但其毕竟指出了研究“*Approximation Proof*”的方向,亦即必须给出 L 和 H 的数学意义,这样才能使得由 L 和 H 构成的逻辑公式也有相应的数学意义。进而,刘清等基于粗糙集理论定义了近似度 λ_* 和 λ^* ,它和基于专业领域的精确数和经验数一起组成粗糙数,并讨论了粗糙逻辑的性质和 $\lambda \in [\lambda_*, \lambda^*]$ 在逻辑公式解释上的价值。另外,刘清于 1996 年底在日本召开的第 5 届国际粗糙集研讨会上提出了一种精度算子粗糙逻辑(AORL),并给出其归结推理的过程。

A.Nakamura 等提出了基于不完全信息系统的粗糙模态逻辑 R1 和 R2。R1 的主要思想是基于不完全信息系统的某种等价关系,他们给出了 R1 的一些特性和决策过程及一个公理化演绎系统,并证明了其完备性和正确性。R2 主要是基于间隔集代数结构,对于 R2 提出了与 R1 不同的不完全信息系统的模态运算的定义,并给出了此逻辑的决策过程,进而,研究了 R2 公理系统的简化。

11.5.4 与其他数学工具的结合

粗糙集与 Fuzzy 集并非是对立的理论,两者既互相区别,又互相补充。从根本上讲,粗糙集体现了集合中对象间的不可区分性,即由于知识的粒度而导致的粗糙性;而 Fuzzy 集则对集合中子类的边界的不清楚定义进行模型化,它体现的是隶属边界的模糊性。它们处理的是两种不同的模糊和不确定性,两者的有机结合可能更好地处理不完全知识。D.Dudios 和 H.Prade 由此提出了粗糙 Fuzzy Set 和 Fuzzy 粗糙 Set 的概念。其主要思想是当等价关系使模糊集合的论域变得粗糙时,定义此模糊集合的相应上近似和下近似;或者把等价关系弱化为模糊相似关系,从而得到一个更具表达力的粗糙模型。并通过相似关系对模糊集合的上近似和下近似的性质进行了详细研究,指明了在不分明性和模糊谓词同时存在的情况下, Fuzzy 粗糙 Set 概念在逻辑推理方面的潜在用途。

D.Dudios 和 H.Prade 同时指出,Shafer 的证据理论¹和 Z. Pawlak 的粗糙集理论是不同术语下的同一个模型。A.Skowron 和 J.Grazymala-Buss 给出了更具体的结论。他们认为,粗糙集理论可以看作证据理论的基础。并在粗糙集理论的框架上重新解释了证据理论的基本概念,特别是用上近似和下近似的术语解释了信念(belief)和似然(plausibility)函数,进而讨论了两者的互补问题。

11.6 粗糙集的实验系统

粗糙集理论已经被证实在实践中是非常有用的,从大量的现实生活中应用的记录来看已经非常明显。这一理论对于 AI 和认知科学尤为重要,在决策支持、专家系统、归纳推理、开关电路等方面有了重要的应用。

近年来,粗糙集理论在数据库中知识发现(KDD)中的应用取得了较大的进展,基于粗糙集理论的方法逐渐成为 KDD 主流方法之一。知识发现或数据库的数据挖掘是 AI 的一个相对新的子领域,它涉及到从不断增长的企业信息数据库中挖掘出额外的非平凡的知识。在这个内容上,主要任务之一是内部数据之间的关联和关系。但是,尽管粗糙集理论对模糊和不完全知识的处理比较出色,但其对原始模糊数据的处理能力较弱。因此和其它方法如模糊数学、神经网络等结合将会取得更好的效果。

基于粗糙集的 KDD 的系统一般都由数据预处理、基于粗糙集或其扩展理论的数据约简、决策算法等部分组成。其大概思想时先进行必要的的数据预处理,为数据约简做准备,然后求出

约简或近似约简，并在此基础上根据值约简等减少属性和个体数目，最终提取规则并将之应用于新对象的分类。

在过去几年中，建立了不少基于粗糙集的 KDD 系统，其中最具有代表性的有 LERS、ROSE、KDD-R、Rough Enough 等。

1. LERS

LERS(Learning from examples based on Rough Set)系统是美国 Kansas 大学开发的基于粗糙集的实例学习系统[Grzyrnala-busse 2000]。它是用 Common Lisp 在 VAX9000 上实现的。LERS 已经为 NASA 的 Johnson 空间中心应用了两年，它是作为一种开发专家系统的工具被引用的，这种类型的专家系统大多数可能被用于空间站释放的板上医疗决策。此外，LERS 还被广泛地用于环境保护、气候研究和医疗研究。

2. ROSE

波兰 Poznan 科技大学基于粗糙集开发了 ROSE(Rough Set data Explorer)，用于决策分析。它是 Rough Das & Rough Class 系统的新版，其中 RoughDas 执行信息系统数据分析任务，RoughClass 支持新对象的分类，这两个系统已经在许多实际领域中得到应用。ROSE 是运行在 PC 兼容机 Windows/NT 上的交互式软件系统。ROSE 的计算模块具有如下特征：

- 数据校验和预处理；
- 采用 Fayyad 和 Irani 离散化算法对连续值进行自动离散化处理；
- 用标准的粗糙集模型或可变精度粗糙集模型对条件属性进行定性估计；
- 用 Romanski 和 Skowron 等人的算法发现属性核及信息表的约简；
- 考察属性对目标分类的相对重要性；
- 选择最重要的属性进行目标分类，删除冗余属性；
- 用 LEM2 算法或 Explore 算法获取决策规则；
- 获取规则的后处理；
- 用决策规则对新目标进行分类；
- 用 K 叠交叉验证方法对决策规则集进行评价。

ROSE 的信息表数据采用 ISF (information system file) 文件格式，是一种纯文本格式。属性分为条件属性和决策属性。

3. KDD—R

KDD-R 是由加拿大的 Regina 大学开发的基于可变精度粗糙集模型，采用知识发现的决策矩阵方法开发了 KDD-R 系统，这个系统被用来对医学数据分析，以此产生症状与病证之间新的联系，另外它还支持电信工业的市场研究。该系统由四部分组成：

- 数据预处理；
- 基于 VPRS 模型的属性依赖分析和消除冗余属性；
- 规则提取；
- 决策。

4. Rough Enough

基于粗糙集理论，挪威 Troll Data Inc. 公司开发了数据挖掘工具 Rough Enough。读者可在网站 <http://www.trolldata.no/renough> 下载该软件。

Rough Enough 系统根据信息系计算得到可辨识矩阵。系统提供许多工具进行集合近似，诸如等价类、决策类、下近似、上近似、边界域、粗糙成员值、泛化决策规则等。采用遗传算法生成约简结果。

11.7 粒度计算

粒度计算是对于利用粒度（群，类，簇）概念来解决问题的理论，方法以及技术的统称。一个粒子通常有不可分辨的，相似的，接近的或者功能的元素组成。信息颗粒化的内涵就是近似。

设 $K=(U,R)$ 为近似空间， R 为 U 的一个区分。则知识 R 的粒度可定义为：

$$GK(R) = (1/|U|^2) * (|R_1|^2 + |R_2|^2 + \dots + |R_m|^2). \quad (11.10)$$

粒度计算是一把大伞，它覆盖了所有有关粒度的理论、方法论、技术和工具的研究。粗略地说，粒度计算是模糊粒度理论的超集，而粗糙集理论和区间计算是粒度数学的子集。当一个问题包括不完备，不确定以及模糊信息时，清楚的区分元素是非常困难的，此时我们必须使用粒度。

粒度计算的概念首先由 Zadeh 教授在“Fuzzy sets and information granularity”(in: *Advances in Fuzzy Set Theory and Applications*) 一文中提出[Zadeh 1979]。Pawlak 等把粒度计算与粗糙集理论联系起来进行研究[Peter 2002]。Yao 提出用决策逻辑语言（DL-语言）来描述集合的粒度（用满足公式 ϕ 元素的集合来定义等价类 $m(\phi)$ ，建立概念之间的 IF-THEN 关系与粒度集合之间的包含关系的联系，并提出利用由所有划分构成的格来求解一致分类问题；提出利用多层粒度化来研究分层粗糙集近似[Yao 2001]。Lin and Yao 利用邻居系统研究粒度计算。

张铃和张钹认为概念可以用子集来表示，不同粒度的概念就体现为不同粒度的子集，一簇概念就构成空间的一个划分——商空间(知识基)，不同的概念簇就构成不同的商空间。故粒度计算，就是研究在给定知识基上的各种子集合之间的关系和转换。商空间的模型用一个三元组来表示，即 (X, F, T) ，其中 X 是论域， F 是属性集， T 是 X 上的拓扑结构。当我们取粗粒度时，即给定一个等价关系 R （或说一个划分），于是我们说得到一个对应于 R 的商集记为 $[X]$ ，它对应于的三元组为 $([X], [F], [T])$ ，称之为对应于 R 的商空间。商空间理论就是研究各商空间之间的关系、各商空间的合成、综合、分解和在商空间中的推理[张铃 2003]。

1962 年，Zeeman 指出认知活动可以被看作是某种函数空间中的相容空间。这种相容空间是通过基于距离函数的相容关系构建的，它被 Zeeman 用于动态系统的稳定性分析中。我们将基于距离关系的相容空间用于信息粒度化的分析。

在不同粒度商描述一个问题的目的就是使得计算机能够解决不同粒度层次上的问题。我们可以用一个相容粒度空间来描述一个问题[Zheng 2005]。一个相容粒度空间 TG 可以形式化为一个三元组 (OS, TR, NTC) ，其中 OS 表示对象集系统，由在相容粒度空间中处理和粒度化的对象组成，它也可以看成是一个对象域； R 表示一个相容关系系统，是一个（参数化的）关系结构，它由一组相容关系组成。它包括一个粒度空间所基于的关系和系数； NTC 表示一个嵌套相容覆盖系统，是一个（参数化的）粒度结构，其中定义了不同层次的粒和基于对象系统和相容关系系统的粒度化过程。它定义了一个嵌套的粒度结构来表示：

- (1) 粒之间，对象之间以及粒和对象之间的关系；
- (2) 粒的组合和分解。

11.7 粗糙集的展望

粗糙集理论已经证明了它在许多实际生活中是完备和十分有用的。粗糙集理论提供了在许多分枝上可应用的有效方法。基于粗糙集理论的粗糙逻辑的研究似乎是值得重视的课题，

因为这种逻辑将使单调逻辑非单调化，从而在 AI 的近似或不精确推理中将发挥不可估量的作用，可见基于粗糙集方法的不精确推理的粗糙逻辑的研究将是十分有前途的。

粗糙集理论的另一项重要课题则是粗糙函数的理论和实践的研究。粗糙函数的各种近似运算，粗糙函数的基本性质，关于它的粗糙连续，粗糙可导，粗糙积分和粗糙稳定性，粗糙函数控制及建立由粗糙实函数控制的离散动态系统等都是典型的问题，这些问题都要求在粗糙函数理论的模型下，给予公式化。这些问题的研究将有贡献于定性推理方法的研究。这些研究实质上是使连续数学离散化。如此，连续数学也能被现代计算机所接受。

基于粗糙集理论的控制也似乎是一个非常有前途的应用领域，而粗糙集理论对神经网络和遗传算法的开发也很重要。如何将粗糙集理论、模糊集理论、证据理论和概率论等不确定的理论用一个统一的逻辑模型来解释也很值得研究。目前，粗糙集理论的研究还有几个领域比较引人注目：

(1)在继承原始粗糙集模型的基本数学性质前提之下，研究如何扩展模型，以更好地用于数据压缩与信息系统分析等。

(2)在分布式粗糙集环境下，不完全的或不确定的知识表示和多 agent 之间知识转换问题。

(3)在特定代数结构上，如何引入上近似与下近似的概念，并研究其数学性质，例如研究概念格结构上的粗糙集运算的定义以及相互关系等。

(4)粗糙集理论与形式语言之间关系的研究等。

从数据库知识发现角度再列举一些可能的研究方向及应用领域。

高效约简算法 高效的约简算法是粗糙集应用于知识发现的基础，现尚不存在一种非常有效方法。因此，寻求快速的约简算法及其增量版本仍然是主要研究方向之一。

大数据集问题 现实中的数据库已经越来越大。粗糙集理论如何应付这一挑战仍旧是一个问题。虽然现在已经有一些有益的探索，但是还没有找到一种令人满意的方法。可能的解决方案有采样、并行化等，更需要发展相应的算法。

多方法融合 现在有许多中数据挖掘方法。实验表明，还没有一种再所有的测试集上表现比其他方法出色。因此多种方法的融合可能是进一步提高分类效率的方法之一。

读者可以在 <http://www.cs.uregina.ca/~roughset> 的 Electronic Bulletin of the Rough Set Community 中看到粗糙集研究的进展。

习 题

1. 简述粗糙集的理论基础是什么。
2. 什么是决策表和决策逻辑，简述决策逻辑语言的基本表达。
3. 解释知识的约简和知识的依赖性。
4. 为什么粗糙集可以进行属性约简，举例说明非一致决策表的约简方法。
5. 最小属性集的含义是什么。
6. 简述基本粗糙集理论存在的主要问题，以及粗糙集理论研究中提出的几种典型扩展模型。
7. 何谓粒度计算？试比较三种粒度计算模型的特点。

第十二章 关联规则

12.1 关联规则挖掘概述

关联规则挖掘的目的是找出数据库中不同数据项集之间隐藏的关联关系，它是近几年研究较多的数据挖掘方法，也是应用最为广泛的数据挖掘方法之一。在数据挖掘的知识模式中，关联规则模式是比较重要的一种。关联规则的概念由 Agrawal、Imielinski、Swami 在 1993 年提出[Agrawal 1993]，是数据中一种简单但很实用的规则。关联规则模式属于描述型模式，发现关联规则的算法属于无监督学习的方法。

目前，关联规则的研究具有以下的发展趋势：一是从单一概念层次关联规则的发现发展到多概念层次的关联规则的发现。也就是说很多具体应用中，挖掘规则可以作用到数据库不同的层面上。例如：在分析超市销售事务数据库过程中，若单从数据库中的原始字段，如面包、牛奶等进行规则挖掘，可能难以发现令人感兴趣的规则。这时如果我们将一些抽象层次的概念也考虑进去，如比面包和牛奶更抽象的概念：食品，则有可能发现新的更为抽象的规则。所以研究在数据库中不同的抽象层次上发现规则和元规则是数据挖掘的新的研究内容。

二是提高算法效率。显然在挖掘规则过程中，需要处理大量的数据库记录，并且可能对数据库进行多次扫描。所以，如何提高算法的效率是非常重要的。共有三种提高效率的思路：一种技术是减少数据库扫描次数，这种技术对效率会有巨大的提高。另一种是利用采样技术，对待挖掘的数据集合进行选择，这在一些效率更为重要的应用中是非常有效的。最后是采用并行数据挖掘。这是因为大规模的数据库经常分布在若干网络节点上，并行挖掘技术显然能提高效率。这对于在 Internet 网上的海量数据挖掘研究具有重要的意义。

此外，对获取的关联规则总规模的控制：如何选择和进一步处理所获得的关联规则；模糊关联规则的获取和发现；高效率的关联规则挖掘算法等也是关联规则要研究的关键性课题。从挖掘的对象上看，由仅在关系数据库中进行挖掘扩充到在文本和 WEB 数据中进行关联的发现等课题也是未来关联规则挖掘要深入研究和解决的问题。

12.1.1 关联规则的意义和度量

关联规则发现的主要对象是事务数据库，其中针对的应用是售货数据，也称为货篮数据（basket data）。如在超级市场的前端收款机中就收集存储了大量的数据。一般情况下，一个事务由如下几个部分组成：事务处理时间，一组顾客购买的物品，物品的数量及金额，以及顾客的标识号（如信用卡号）。现实中，这样的例子很多。利用前端收款机收集存储了大量的数据。

在事务数据库中，让我们考察一些涉及到许多物品的事务(transaction)：事务 1 中出现了物品甲，事务 2 中出现了物品乙，事务 3 中则同时出现了物品甲和乙。那么，物品甲和乙在事务中的出现相互之间是否有规律可循呢？在数据库的知识发现中，关联规则就是描述这种在一个事务中物品之间同时出现的规律的知识模式。更确切的说，关联规则通过量化的数字描述物品甲的出现对物品乙的出现有多大的影响。

这些数据中常常隐含着如下形式的关联规则：在购买面包的顾客当中，有 70% 的人同时购买了黄油。这些关联规则具有一定的商业价值。如商场管理人员可以根据这些关联规则更

好地规划商场，如把面包和黄油这样的商品摆放在一起，能够促进销售。

有些数据不像售货数据那样很容易就能看出一个事务是许多物品的集合，但其本质上仍然可以像对售货数据一样处理。比如人寿保险，一份保单就是一个事务。保险公司在接受保险前，往往需要记录投保人详尽的信息，有时还要到医院做身体检查。保单上记录有投保人的年龄、性别、健康状况、工作单位、工作地址、工资水平等。这些投保人的个人信息就可以看作事务中的物品。通过分析这些数据，可以得到类似以下这样的关联规则：年龄在 40 岁以上，工作在 A 区的投保人当中，有 45% 的人曾经向保险公司索赔过。在这条规则中，“年龄在 40 岁以上”是物品甲，“工作在 A 区”是物品乙，“向保险公司索赔过”则是物品丙。可以看出来，A 区可能污染比较严重，环境比较差，导致工作在该区的人健康状况不好，索赔率也相对比较高。

设 $R=\{I_1, I_2, \dots, I_m\}$ 是一组物品集， W 是一组事务集。 W 中的每个事务 T 是一组物品， $T \subset R$ 。假设有一个物品集 A ，一个事务 T ，如果 $A \subset T$ ，则称事务 T 支持物品集 A 。关联规则是如下形式的一种蕴含： $A \rightarrow B$ ，其中 A 、 B 是两组物品， $A \subset I$ ， $B \subset I$ ，且 $A \cap B = \emptyset$ 。一般可以采用四个参数来描述一个关联规则的属性：

(1) 可信度 (Confidence)

设 W 中支持物品集 A 的事务中，有 $c\%$ 的事务同时也支持物品集 B ， $c\%$ 称为关联规则 $A \rightarrow B$ 的可信度。简单地说，可信度就是指在出现了物品集 A 的事务 T 中，物品集 B 也同时出现的概率有多大。如上面所举的面包和黄油例子，该关联规则的可信度就回答了这样一个问题：如果一个顾客购买了面包，那么他也购买黄油的可能性有多大呢？在上述例子中，购买面包的顾客中有 70% 的人购买了黄油，所以可信度是 70%。

(2) 支持度 (Support)

设 W 中有 $s\%$ 的事务同时支持物品集 A 和 B ， $s\%$ 称为关联规则 $A \rightarrow B$ 的支持度。支持度描述了 A 和 B 这两个物品集的并集 C 在所有的事务中出现的概率有多大。如果某天共有 1000 个顾客到商场购买物品，其中有 100 个顾客同时购买了面包和黄油，那么上述的关联规则的支持度就是 10% (100/1000)。

(3) 期望可信度 (Expected confidence)

设 W 中有 $e\%$ 的事务支持物品集 B ， $e\%$ 称为关联规则 $A \rightarrow B$ 的期望可信度。期望可信度描述了在没有任何条件影响时，物品集 B 在所有事务中出现的概率有多大。如果某天共有 1000 个顾客到商场购买物品，其中有 200 个顾客购买了黄油，则上述的关联规则的期望可信度就是 20%。

(4) 作用度 (Lift)

作用度是可信度与期望可信度的比值。作用度描述物品集 A 的出现对物品集 B 的出现有多大的影响。因为物品集 B 在所有事务中出现的概率是期望可信度；而物品集 B 在有物品集 A 出现的事务中出现的概率是可信度，通过可信度对期望可信度的比值反映了在加入“物品集 A 出现”的这个条件后，物品集 B 的出现概率发生了多大的变化。在上例中作用度就是 $70\% / 20\% = 3.5$ 。

用 $P(A)$ 表示事务中出现物品集 A 的概率， $P(B|A)$ 表示在出现物品集 A 的事务中，出现物品集 B 的概率，则以上四个参数可用公式表示，如表 12.1 所示。

可信度是对关联规则的准确度的衡量，支持度是对关联规则重要性（或适用范围）的衡量。支持度说明了这条规则在所有事务中有多大的代表性，显然支持度越大，关联规则越重要，应用越广泛。有些关联规则可信度虽然很高，但支持度却很低，说明该关联规则实用的机会很小，因此也不重要。

表 12.1 四个参数的计算公式

名称	描述	公式
可信度(C Confidence)	在物品集 A 出现的前提下, B 出现的概率	$P(B A)$
支持度(Support)	物品集 A、B 同时出现的概率	$P(A \cap B)$
期望可信度 (Expected Confidence)	物品集 B 出现的概率	$P(B)$
作用度(Lift)	可信度对期望可信度的比值	$P(B A)/P(B)$

期望可信度描述了在没有物品集 A 的作用下, 物品集 B 本身的支持度; 作用度描述了物品集 A 对物品集 B 的影响力的大小。作用度越大, 说明物品集 B 受物品集 A 的影响越大。一般情况, 有用的关联规则的作用度都应该大于 1, 只有关联规则的可信度大于期望可信度, 才说明 A 的出现对 B 的出现有促进作用, 也说明了它们之间某种程度的相关性, 如果作用度不大于 1, 则此关联规则也就没有意义了。

应该指出: 在这四种度量中, 最常用的是可信度和支持度。

12.1.2 经典的挖掘算法

关联规则的挖掘问题就是在事务数据库 D 中找出具有用户给定的最小支持度 minsup 和最小可信度 minconf 的关联规则。

挖掘关联规则问题可以分解为以下两个子问题:

①找出存在于事务数据库中的所有大物品集(常用物品集、或频繁集)。物品集 X 的支持度 support(X)不小于用户给定的最小支持度 minsup, 则称 X 为大物品集(large itemset)。

②利用大项集生成关联规则。对于每个大项集 A, 若 $B \subset A, B \neq \emptyset$, 且 $\text{Confidence}(B \Rightarrow (A-B)) \geq \text{minconf}$, 则构成关联规则 $B \Rightarrow (A-B)$ 。

第②个子问题比较容易。目前大多数研究集中在第一个子问题上。以下我们对算法的介绍也集中在第一个子问题上。

下面我们先介绍几个经典的关联规则挖掘算法, 即 Apriori 算法[Agrawal 93]、抽样算法、DIC 算法。著名的 Apriori 算法主要工作在于寻找大物品集, 它利用了大物品集向下封闭性, 即大物品集的子集必须是大物品集, 它是宽度优先算法。先计算所有的 1-项集(k-项集是含有 k 个项的项集), 记为 C_1 。找出所有的常用 1-项集, 记为 L_1 。然后根据常用 1-项集确定候选 2-项集的集合, 记为 C_2 。从 C_2 找出所有的常用 2-项集, 记为 L_2 。然后根据常用 2-项集确定候选 3-项集的集合, 记为 C_3 。从 C_3 找出所有的常用 3-项集, 记为 L_3 。如此下去直到不再有候选项集。

算法 12.1 Apriori 算法:

Input: DB, minsup.

Output: Result=所有的频繁项集, 和它们的支持度。

方法:

Result:={ };

k: =1;

C_1 : =所有的 1-项集


```

while( $C_k$ )do
begin
  为每一个  $C_k$  中的项集生成一个计数器;
  for( $i=1;i \leq |DB|;i++$ )
    begin /*所有 DB 中的记录  $T^*$ */
      对第  $i$  个记录  $T$  支持的每一个  $C_k$  中的项集, 其计数器加一;
    end
   $L_k := C_k$  中满足大于 minsup 的全体项集;
   $L_k$  的支持度保留;
  Result:=Result  $\cup$   $L_k$ ;
   $C_{k+1} :=$  所有的( $k+1$ )-项集中满足其  $k$ -子集都在  $L_k$  里的全体;
   $k=k+1$ ;
enddo

```

Apriori 算法扫描 DB 多遍, 第 k 遍计算 k -项集。如果顶层项集中元素个数最多的为 K , 则该算法扫描 DB 至少 K 遍, 也可能 $K+1$ 遍。对 Apriori 算法的改进主要利用 hash[DHP]的方法, 它通过减小候选项集的个数、减小记录长度、减少记录总数的方法, 实现减少验证记录 T 支持 C_k 的计算。抽样算法需要负边界的概念。它也利用了频繁项集向下封闭性。给定一个向下封闭的项集的集合 SI 的幂集合, 如果一个项集的所有子集包含于 S , 而它自身不在 S 中, 则它是负边界的一个元素。所有这样的元素构成负边界。记为 $Bd(S)$ 。例如 $I=\{A,B,C,D,E\}$, $S=\{\{A\},\{B\},\{C\},\{E\},\{A,B\},\{A,C\},\{A,E\},\{C,E\},\{A,C,E\}\}$, 则 $Bd(S)=\{\{B,C\},\{B,E\},\{D\}\}$ 。如果 $Bd(S)$ 中都是非频繁项集, 则 S 是频繁项集的超集。

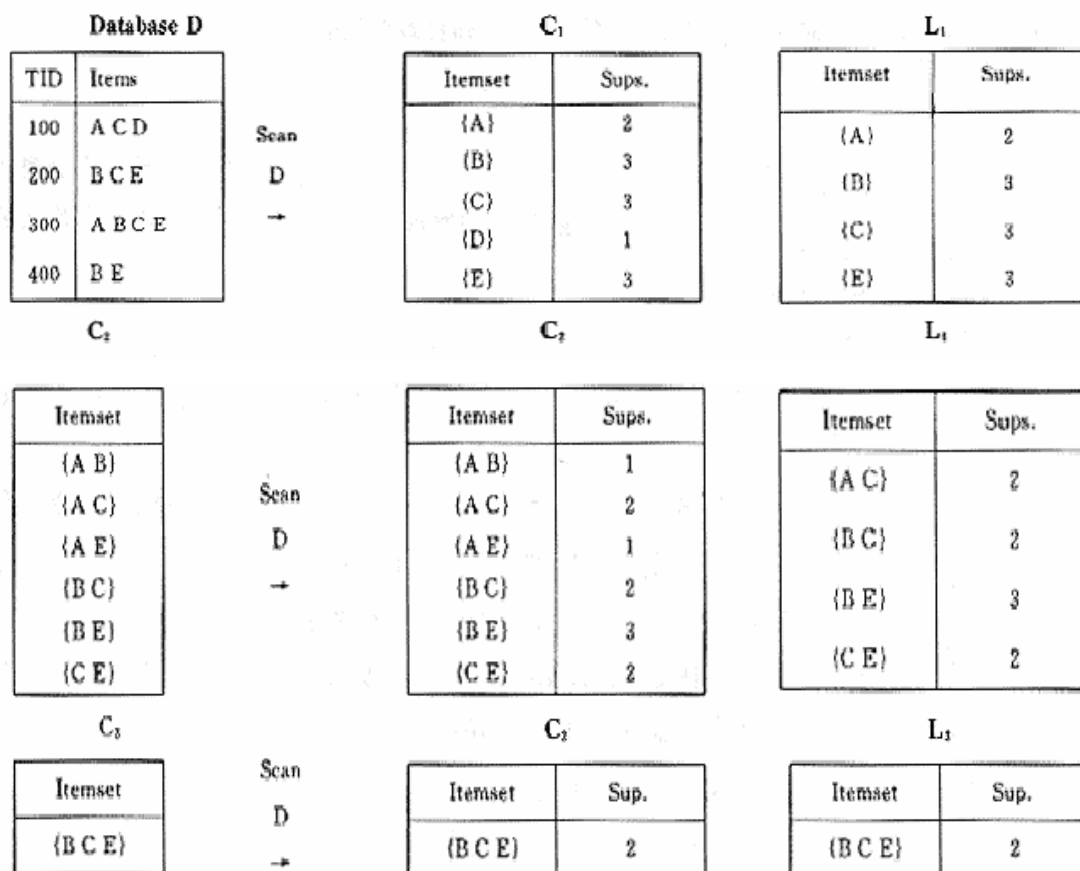
对于数据库 D , 如图 12.1 所示, 在第一遍扫描数据库过程中, 通过简单地扫描整个事务集中每个数据项发生的次数, 得到候选数据项集 C_1 , 假设给定的最小支持度为 2, 则可得到一维数据项集 L_1 。为了生成 L_2 , 注意到大数据项集的任何子集也具有最小支持度, Apriori 算法用运算 $L_1 * L_1$ 产生数据项集 L_2 , 这里, 运算 $*$ 定义为

$$L_1 * L_2 = \{X \cup Y | X, Y \in L_k | X \cap Y = k-1\}$$

由此得到候选集 C_2 , 再由最小支持度得到 L_2 。从 L_2 在生成 C_3 时, 首先两个具有相同首项的数据项: $\{BC\}$ 和 $\{BE\}$ 可以确定下来。再考察 $\{BC\}$ 和 $\{BE\}$ 的尾项生成的数据项集 $\{CE\}$ 是否满足最小支持度, 结果是成立的。这样, $\{BCE\}$ 的所有二维子集都是大数据项, 所以 $\{BCE\}$ 为候选数据项。同时, 从 L_2 也得不到其它三维候选数据项。这样 C_3 就确定了, 同理求出 L_3 。到此为止, 我们也得不到更高维的数据项集了。所以, 整个大数据项集就确定了。

抽样算法的主要思想是取一个比 minsup 还小的阈值 lowsup, 在抽样数据 db 上用 Apriori 算法求出 $Support(db(X)) > lowsup$ 的项集的集合, 记为 S , 假设 S 是常用物品集的超集。求出负边界 $Bd(S)$, 扫描 DB, 计算 S 和 $Bd(S)$ 的支持度, 如果 $Bd(S)$ 中都是非常用物品集, 则 S 是常用物品集的超集。否则报告失败, 加 $Bd(S)$ 中的常用物品集到 S 中, 再求出负边界 $Bd(S)$, 再扫描 DB, 再验证, 直到 S 不能增加而结束。

图 3.1 候选集和数据项集的生成



DIC 算法是将 DB 分为 M 块，在第一块执行 Apriori 算法的第一遍，在第二块执行 Apriori 算法的第一遍和第二遍，在第三块执行 Apriori 算法的第一、第二和第三遍，如此下去直到 DB 的结尾。回绕 DB,在第一块执行 Apriori 算法的第二遍以后的计算，在第二块执行 Apriori 算法的第三遍以后的计算，如此下去直到全部常用物品集被确定。另外在计算第一块之后有一个常用物品集和候选项集动态调整的维护过程。DIC 算法还有一个随机抽样的版本。

12.2 广义模糊关联规则的挖掘

对于挖掘数量属性的关联规则，常用的方法是将连续数据离散化，从而把数量属性的关联规则的问题转换成布尔型关联规则的问题进行讨论。一种方法是将属性的论域划分成不重叠的区间，再将数据库中的离散数据映射到这些区间中，由于明显的区间划分会将某些区间附近的一些潜在元素排斥在外，从而导致一些有意义的区间可能被忽略掉。另一种方法是将属性的论域划分成重叠的区间，这时处于边界附近的元素就有可能同时处于两个区间，由于这些元素同时对两个区间都作贡献，就有可能造成过分强调这些元素的作用，从而导致某些区间的意义也被过分地强调了。上述两种方法的缺点主要是由于边界划分过硬。为了解决这个问题，陆建江等提出可以采用定义在属性论域上的模糊集来软化边界[陆建江 00]。这是因为模糊集可以在集合元素和非集合元素之间提供非常平滑的变迁。有了平滑的变迁，几乎所有边界附近的元素不会再被排斥在外，同时也不会被过分地强调。属性论域上模糊集的元素

隶属度取为语言值，语言值将采用 R 上的有界闭的正模糊数和零模糊数来表示。这样，挖掘数量属性关联规则就转换成挖掘广义模糊关联规则的问题。按照[马洪文 00]，可以给出下述定义：

定义 12.1 设 R 为实数域，称闭区间 $[a,b]$ 为闭区间数，其中 $a,b \in R, a \leq b$ 。

定义 12.2 设 $[a,b], [c,d]$ 为两个闭区间数，有如下定义，其中 $0 \notin [c,d]$ ：

$$\begin{aligned} [a,b] + [c,d] &= [a+c, b+d]; \\ [a,b] - [c,d] &= [a-d, b-c]; \\ [a,b] \times [c,d] &= [ac \wedge ad \wedge bc \wedge bd, ac \vee ad \wedge bc \wedge bd]; \\ [a,b] \div [c,d] &= \left[\frac{a}{c} \wedge \frac{a}{d} \wedge \frac{b}{c} \wedge \frac{b}{d}, \frac{a}{c} \vee \frac{a}{d} \vee \frac{b}{c} \vee \frac{b}{d} \right]; \end{aligned}$$

定义 12.3 设 A 是 R 上的模糊集，

- 1) A 称为 R 上闭凸模糊集，当且仅当 $\forall \lambda \in (0,1]$ ， A_λ 是闭凸集，即 A_λ 是闭区间；
- 2) A 称为 R 上的正则模糊集，当且仅当 $\exists x_0 \in R$ ，使 $A(x_0)=1$ ，这时把 x_0 称为 A 的正则点；
- 3) 若对 $\forall \lambda \in (0,1]$ ， A_λ 为有界集，则称 A 为有界模糊集；
- 4) R 上正则凸模糊集 A 称为一个模糊数；正则闭凸模糊集称为闭模糊数；正则有限闭凸模

糊集称为有界闭凸模糊数， $\bar{\theta}$ 为零模糊数，其中 $\bar{\theta} = \begin{cases} 1, x = 0 \\ 0, x \neq 0 \end{cases}$ ；

- 5) 设 A 为模糊数，若 $\text{supp}A = \{x \in R \mid A(x) > 0\}$ 所含数都是正实数，则称 A 为正模糊数。记有界闭的正模糊数的全体为 G ，记 $\tilde{G} = G \cup \{\bar{\theta}\}$

定义 12.4 在 \tilde{G} 中定义“ \leq ”如下： $\forall A, B \in \tilde{G}$ ， $A \leq B$ 当且仅当对 $\forall \lambda \in (0,1]$ ， $a_1^\lambda \leq b_1^\lambda$ 且 $a_2^\lambda \leq b_2^\lambda$ ；其中 $A_\lambda = [a_1^\lambda, a_2^\lambda]$ ， $B_\lambda = [b_1^\lambda, b_2^\lambda]$ 。易知“ \leq ”是 \tilde{G} 中的一个偏序。

定义 12.5 设 $A, B \in \tilde{G}$ ，定义：

$$\begin{aligned} (A+B)(z) &= \bigvee_{x+y=z} (A(x) \wedge B(y)), \forall z \in R \\ (A-B)(z) &= \bigvee_{x-y=z} (A(x) \wedge B(y)), \forall z \in R \\ (A \times B)(z) &= \bigvee_{x \times y=z} (A(x) \wedge B(y)), \forall z \in R \\ (A \div B)(z) &= \bigvee_{x \div y=z} (A(x) \wedge B(y)), \forall z \in R \end{aligned}$$

$$(kA)(z) = A\left(\frac{z}{k}\right), k \neq 0, \forall z \in R$$

定理 12.1 设 $A, B \in \tilde{G}$, 则 $\forall \lambda \in (0, 1]$, 有 $(A \pm B)_\lambda = A_\lambda \pm B_\lambda; (A \times B)_\lambda = A_\lambda \times B_\lambda; A / B_\lambda = A_\lambda / B_\lambda$,

$$B \neq \bar{\theta}, (kA)_\lambda = kA_\lambda, k \neq 0$$

注 1: 由定理 12.1 及定义 12.2 易知 $A, B \in \tilde{G}$, 有 $A + B \in \tilde{G}$, $A \times B \in \tilde{G}$,

$$A / B \in \tilde{G} (B \neq \bar{\theta}), \quad kA \in \tilde{G} (k > 0)$$

下面讨论广义模糊关联规则的计算方法。设 $T = \{t_1, t_2, \dots, t_n\}$ 是一个数据库, t_i 表示 T 的第 i 个元组, $I = \{i_1, i_2, \dots, i_n\}$ 表示属性集, $t_j[i_k]$ 表示属性 i_k 在第 j 个元组上的值。设 $X = \{x_1, x_2, \dots, x_p\}$, $Y = \{y_1, y_2, \dots, y_q\}$ 是 I 的子集, 且 $X \cap Y = \Phi$, $D = \{f_{x1}, f_{x2}, \dots, f_{xp}\}$, $E = \{f_{y1}, f_{y2}, \dots, f_{yq}\}$, 其中 $f_{xi} (i=1, 2, \dots, p)$ 和 $f_{yj} (j=1, 2, \dots, q)$ 分别是定义在属性 x_i 和 y_j 的论域上的模糊集, 这些模糊集中元素的隶属度取为语言值, 语言值用有界闭的正模糊数和零模糊数表示。给定阈值 ε' , 最小支持率 α' , 最小信任度 β' , 这里 $\alpha', \beta', \varepsilon'$ 都是有界闭的正模糊数。所要讨论的广义模糊关联规则的形式为“如果 X 是 D 则 Y 是 E ”。下面将分两步来讨论此规则。

- 1) 令 $f_{xj}(t_i[x_j]) = x'_{ij}$, $i=1, 2, \dots, n$; $j=1, 2, \dots, p$; $f_{yj}(t_i[y_j]) = y'_{ij}$, $i=1, 2, \dots, n$; $j=1, 2, \dots, q$; 这里 x'_{ij} 与 y'_{ij} 都是有界闭的正模糊数或零模糊数。取

$$\bar{x}'_{ij} = \max\{x \in R \mid x'_{ij}(x) = 1\} \quad i=1, 2, \dots, n; j=1, 2, \dots, p;$$

$$\bar{y}'_{ij} = \max\{x \in R \mid y'_{ij}(x) = 1\} \quad i=1, 2, \dots, n; j=1, 2, \dots, q;$$

$$\bar{\alpha}' = \max\{x \in R \mid \alpha'(x) = 1\} \quad \bar{\beta}' = \max\{x \in R \mid \beta'(x) = 1\}$$

$$\bar{\varepsilon}' = \max\{x \in R \mid \varepsilon'(x) = 1\} \quad M = \max\{\bar{x}'_{ij}, \bar{y}'_{ij}, \bar{\alpha}', \bar{\beta}', \bar{\varepsilon}'\}$$

$$x_{ij} = \frac{x'_{ij}}{M}, i=1, 2, \dots, n; j=1, 2, \dots, p \quad y_{ij} = \frac{y'_{ij}}{M}, i=1, 2, \dots, n; j=1, 2, \dots, q$$

$$\alpha = \frac{\alpha'}{M}, \beta = \frac{\beta'}{M}, \gamma = \frac{\gamma'}{M}$$

易知 $X_{ij}, Y_{ij}, \alpha, \beta, \varepsilon$ 仍都是有界闭的正模糊数或零模糊数, 且它们的正则点都落在 $[0, 1]$ 区间上。

- 2) 首先给出两个定义。

定义 12.6 广义模糊关联规则“如果 X 是 D 则 Y 是 E ”的广义支持率记为 S , 这里

$$S = \frac{\sum_{i=1}^n [\prod_{j=1}^p \bar{\alpha}(x_{ij}) \times \prod_{j=1}^q \bar{\alpha}(y_{ij})]}{n}, \quad \bar{\alpha}(x) = \begin{cases} x, & x \geq \varepsilon' \\ \bar{\theta}, & \text{其它} \end{cases}$$

定义 12.7 广义模糊关联规则“如果 X 是 D 则 Y 是 E”的广义信任度记为 C,这里

$$C = \frac{S}{\frac{1}{n} \sum_{i=1}^n [\prod_{j=1}^p \bar{a}(x_{ij})]} , \quad \bar{a}(x_{ij}) = \begin{cases} x_{ij}, x_{ij} \geq \varepsilon' \\ \bar{\theta}, \text{其它} \end{cases}$$

当 $\frac{1}{n} \sum_{i=1}^n [\prod_{j=1}^p \bar{a}(x_{ij})] = \bar{\theta}$ 时, 易知 $S = \bar{\theta}$, 故规则肯定不会被采用。不妨设

$\frac{1}{n} \sum_{i=1}^n [\prod_{j=1}^p \bar{a}(x_{ij})] \neq \bar{\theta}$, 此时易知定义 12.6 中的 S 和定义 12.7 中的 C 都属于 \tilde{G} 。同时,由第一步

知 X_{ij} 的正则点都落在 $[0,1]$ 区间上, 故 $\frac{1}{n} \sum_{i=1}^n [\prod_{j=1}^p \bar{a}(x_{ij})]$ 的正则点也落在 $[0,1]$ 区间上, 即

C 的最大正则点不会小于 S 的最大正则点, 也就是说 C 不会小于 S。由于 S, C, α , β 都属于 \tilde{G} , 故 S 和 α , C 和 β 都可以比较, 当 $S \geq \alpha$ 且 $S \geq \beta$ 时, 则认为规则“若 X 是 D 则 Y 是 E”可被采用。

12.3 挖掘关联规则的数组方法

文献[马洪文 00]提出了一个新的计算候选项集出现频率的方法, 即效率较高的数组方法。

扫描 DB, 求出频率 1-项集的集合 $F_1 = \{a_1, a_2, \dots, a_n\}$, 称 n 维数组 (b_1, b_2, \dots, b_n) , $b_i \in F_1$ 为最大空间, 再扫描 DB, 求出最大空间上的支持度。最多扫描 DB 两遍。算法定义如下:

算法 12.2 数组方法。

Input: DB, minsup.

Output: Result=所有的频繁项集, 和它们的支持度。

方法:

```
/* 第一遍整个扫描 DB */
求出频繁 1-项集的集合  $F_1 = \{a_1, a_2, \dots, a_n\}$ ;
求出最大空间 V;
if (V 能完全装入内存)
    /* 第二遍整个扫描 DB */
    求出最大空间上的支持度
else
    V 分为若干能完全装入内存的子块
    /* 第二遍整个扫描 DB */
    for (V 的每一块)
```

求出最大空间子块上的支持度;

endif

Result: =V 上的支持度大于 minsup 的项集;

首先对比一下该算法与抽样算法。由于抽样算法为了保证基本上扫描一遍成功,所以选取的 lowsup 比 minsup 小很多,产生的 S 元素个数一般情况下比 L_k 、 C_k 元素个数大很多,有时是几倍。这大大降低了它的效率,而减小 I/O 时间带来的益处也消耗没了。

其次对比一下该算法与 DIC 算法。由于 DIC 算法实质上类似于新算法。它在抽样时选取前 K 块做为子样,所以其产生的频繁项集的超集 S,一定不如随机抽样时产生的频繁项集的超集 S 更接近实际数据。如果数据分布不均匀时, DIC 算法将比 Apriori 算法还要费时。

而且计算 S 支持度采取数组计算的方法,它的效率是很高的。例如,假设有一个 8 个元素的顶层,有一条记录含有顶层项集中的 4 项。当扫描 DB 时,对于 Apriori 算法,用 hash 树计算至少要检验=15 次,计数 15 次。而采用数组计算的方法,则只需检验 8 次,效率提高 50%。

12.4 任意多表间关联规则的并行挖掘

目前数据挖掘基本是在单个表(关系)内进行的,文献[74]首次将 AQ 学习算法用于多表间关联规则的提取,但由于 AQ 算法本身的制约,表间的联系仅能通过关键字表达,数据必须全部放入内存,因而局限性较大。由于表间连接时空开销较大,而且多个表可能属于不同的数据库,甚至不同的机构,因安全性等因素不能将其在物理上连接起来,因此产生从多个表中直接提取关联规则的问题。

文献[左万利 99]考虑任意多表(关系)间以语义相关属性所表示的一般联系。相关属性可以是关键字或其它对等属性。算法面向大数据集,不受内存容量的制约。数据限定为布尔属性,类别属性数据转换为布尔属性数据后也可用此方法。数据挖掘首先在多表内并行进行,然后根据相关属性推算多关系间的规则,得到满足给定支持度和置信度的所有跨表间的关联规则。

12.4.1 问题的形式描述

设表(关系)为 $\psi_1, \psi_2, \dots, \psi_n$, $\text{Attr}(\psi_i)$ 表示 ψ_i ($i=1,2,\dots,n$) 的属性集, $\text{Ac}_i = \text{Attr}(\psi_i) \cap \text{Attr}(\psi_{i+1})$ 为 ψ_i 与 ψ_{i+1} ($i=1,2,\dots,n-1$) 的公共属性集。令 $A_1 = \text{Attr}(\psi_1) - \text{Ac}_1$, $A_i = \text{Attr}(\psi_i) - \text{Ac}_{i-1} - \text{Ac}_i$ ($i=2, 3, \dots, n-1$), $A_n = \text{Attr}(\psi_n) - \text{Ac}_{n-1}$, 将 A_i 的所有属性编号为 $1, 2, \dots, r_i$ 。为处理方便,将表(关系)转换为事物形式,若 $\psi = \{t_1, t_2, \dots, t_l\}$ 是元组的集合,则 $\delta = \{t'_1, t'_2, \dots, t'_l\}$ 是对应事物的集合,其中 $t'_k = \{i | t_k[i] = \text{True}, t_k \in \psi\}$ 。令 $\delta_i = \{t'_{i1}, t'_{i2}, \dots\}$ 是由 $\psi_i = \{t_{i1}, t_{i2}, \dots\}$ 得到的事物集合。任意多表间关联规则数据挖掘就是寻找所有满足给定支持度 minsup 和置信度 minconf 的蕴含式:

$$X \Rightarrow Y, X, Y \subset \bigcup_i A_i, \text{ 且 } X \cap Y = \emptyset, X \cup Y \subseteq A_i, i=1, 2, \dots, n$$

其中 X, Y 是属性(编号)的集合,称为项集。这种规则的直观含义:在 $\psi_1, \psi_2, \dots, \psi_n$ 按公共属性 $\bigcup_i \text{Ac}_i=1$ 连接得到的表(关系)中,包含 X 的元组通常也包含 Y。令 m_1 表示 ψ_1 中包含公共属性 ($\text{Ac}_1=1$) 的元组的个数, m_n 表示 ψ_n 中包含公共属性 ($\text{Ac}_{n-1}=1$) 的元组的个数, m_i 表示 ψ_i 中包含公共属性 ($\text{Ac}_{i-1}=1$ 且 $\text{Ac}_i=1, i=2, \dots, n-1$) 的元组的个数,则 $\psi_1, \psi_2, \dots, \psi_n$ 按公共

属性 $\bigcup_i A_i=1$ 连接后所得表中元组的个数 $joinsize=m_1 \times m_2 \times \dots \times m_n$ 。令 $X=X_1 \cup X_2 \cup \dots \cup X_n$ ，其中 $X_i \in A_i$ 。若 $count(X_i)$ 表示 δ_i 中包含 X_i 且 $Attr(\psi_i)-A_i=1$ 的元组的个数(如 $X_i=\emptyset$ ，则 $count(X_i)=m_i$)，并设 $e = \prod_{i=1}^n count(X_i \cup Y_i)$ ，则 $X \Leftrightarrow Y$ 的支持度为

$$sup(X \Rightarrow Y) = sup(Y \Rightarrow X) = \frac{e}{joinsize}$$

置信度为

$$conf(X \Rightarrow Y) = \frac{e}{\prod_{i=1}^n count(X_i)}, \quad conf(Y \Rightarrow X) = \frac{e}{\prod_{i=1}^n count(Y_i)}$$

12.4.2 单表内大项集的并行计算

假定事物及项集中的项按字典排序，含有 k 个项的项集称为 k 项集， k 项集 c 的 k 个项表记为 $c[1], c[2], \dots, c[k]$ 。设 u 是关于 A_i 的一个项集，由于只对多表间关联规则感兴趣，仅当存在一个关于 $A_j(j \neq i)$ 的项集 v ，使得 $count(u) \times count(v) / joinsize \geq minsup$ 时， u 才称为大项集。反之，若对任意关于 $A_j(j \neq i)$ 的项集 v ，有 $count(u) \times count(v) / joinsize < minsup$ ，则 u 不是大项集。含有 k 个项的大项集称为大 k 项集，所有大 k 项集的集合记为 L_k 。若 u 是一个大项集，则 u 的任意一个非空子集是大项集；反之，若 u 的某一非空子集不是大项集，则 u 不是大项集。如果 u 不是大项集，则它与任意关于 $\bigcup_{j \neq i} A_j$ 的项集 v 不能产生多表间的大项集，因而不能参与

形成多关系间规则。为避免产生不必要的 k 项集，首先生成 L_k 的一个较小的超集 C_k ， C_k 中任意一个 k 项集的所有非空子集均为大项集，称 C_k 为候选大 k 项集的集合，由 L_{k-1} 得到，算法 `candi_gen` 见文献 [244]。

由 C_{ik} (其中 i 表示关系 ψ_i ， k 表示 k 项集) 求 L_{ik} 需扫描 δ_i 并统计 C_{ik} 中各 k 项集在 δ_i 中出现的次数，此工作由 `count` 完成。对于每个 $t \in \delta_i, c \in C_{ik}$ ，若 $c \subseteq t$ ，则 $c.count = c.count + 1$ 。 C_{i1} 分别与各自对应的关系有关，因而可同时计算。 L_{i1} 与 C_{i1} 有关，需待 C_{i1} 求出后得到。一旦 L_{i1} 求得后， (C_{ik}, L_{ik}) 与 (C_{jk}, L_{jk}) ($k \geq 2, i \neq j$) 的计算互不依赖，可以同时进行。

在 $\delta_1, \delta_2, \dots, \delta_n$ 中同时求大项集 $L_{11}, L_{12}, \dots, L_{21}, L_{22}, \dots, L_{n1}, L_{n2}, \dots$ 的并行算法：

算法 12.3 求大项集的并行算法。

Parbegin

```
{C11={all 1-itemset in A1};
  forall transaction t1 ∈ δ1 do count(t1,C11);
{C21={all 1-itemset in A2};
  forall transaction t2 ∈ δ2 do count(t2,C21);
...
{Cn1={all 1-itemset in An};
```

```

    forall transaction  $t_n \in \delta_n$  do count( $t_n, C_{n1}$ );
  parend;
  parbegin
     $L_{11} = \{c_1 \mid c_1 \in C_{11}, (\exists c \in C_{j1}) \wedge (j \neq 1), (c_1.count \times c.count) / (m_1 \times m_j) \geq \text{minsup}\}$ ;
     $L_{21} = \{c_2 \mid c_2 \in C_{21}, (\exists c \in C_{j1}) \wedge (j \neq 2), (c_2.count \times c.count) / (m_2 \times m_j) \geq \text{minsup}\}$ ;
    ...
     $L_{n1} = \{c_n \mid c_n \in C_{n1}, (\exists c \in C_{j1}) \wedge (j \neq n), (c_n.count \times c.count) / (m_n \times m_j) \geq \text{minsup}\}$ ;
  parend;
  parbegin
    {for ( $k=2$ ;  $L_{1k-1} \neq \emptyset$ ;  $k++$ )
      {candi_gen( $L_{1k-1}, C_{1k}$ );
        forall transaction  $t_1 \in \delta_1$  do count( $t_1, C_{1k}$ );
         $L_{1k} = \{c_1 \mid c_1 \in C_{1k}, (\exists c \in C_{j1}) \wedge (j \neq 1), (c_1.count \times c.count) / (m_1 \times m_j) \geq \text{minsup}\}$ };
      }
    {for ( $k=2$ ;  $L_{2k-1} \neq \emptyset$ ;  $k++$ )
      {candi_gen( $L_{2k-1}, C_{2k}$ );
        forall transaction  $t_2 \in \delta_2$  do count( $t_2, C_{2k}$ );
         $L_{2k} = \{c_2 \mid c_2 \in C_{2k}, (\exists c \in C_{j1}) \wedge (j \neq 2), (c_2.count \times c.count) / (m_2 \times m_j) \geq \text{minsup}\}$ };
      }
    ...
    {for ( $k=2$ ;  $L_{nk-1} \neq \emptyset$ ;  $k++$ )
      {candi_gen( $L_{nk-1}, C_{nk}$ );
        forall transaction  $t_n \in \delta_n$  do count( $t_n, C_{nk}$ );
         $L_{nk} = \{c_n \mid c_n \in C_{nk}, (\exists c \in C_{j1}) \wedge (j \neq n), (c_n.count \times c.count) / (m_n \times m_j) \geq \text{minsup}\}$ };
      }
  parend;
  Answer1 =  $\bigcup_k L_{1k}$ ;
  Answer2 =  $\bigcup_k L_{2k}$ ;
  ...
  Answern =  $\bigcup_k L_{nk}$ .

```

若在网络环境中, $\psi_1, \psi_2, \dots, \psi_n$ 分别属于不同的站点, 当 $L_{11}, L_{21}, \dots, L_{n1}$ 在各自站点求出后传给其它站点, 则 $L_{1k}, L_{2k}, \dots, L_{nk} (k \geq 2)$ 的计算可以在不同站点上同时进行。

12.4.3 任意多表间大项集的生成

由上述多关系间大项集的定义, 由单关系内的大项集生成两关系间的大项集, 进而生成三关系间的大项集乃至多关系间所有可能的大项集, 多关系间大项集的生成算法为

算法 12.4 多关系间大项集的生成算法。

```

Parbegin
  forall large k-itemset  $u \in L_{1k}, k \geq 1$  do  $u.sup = u.count / m_1$ ;
  forall large k-itemset  $u \in L_{2k}, k \geq 1$  do  $u.sup = u.count / m_2$ ;
  ...
  forall large k-itemset  $u \in L_{nk}, k \geq 1$  do  $u.sup = u.count / m_n$ ;

```



```

parend;
multitemset1 =  $\bigcup_i \text{Answer}_i$ ;

for (i=2; multitemseti-1 ≠ ∅; i++) do
    forall large itemset u ∈ multitemseti-1 do
        forall large itemset v ∈ multitemset1 do
            if (u ∩ v = ∅) and ((u.sup × v.sup) ≥ minsup) then
                {(u+v).sup = u.sup × v.sup;
                 insert u+v into multitemseti;}
multitemset = multitemset2 ∪ multitemset3 ∪ ...

```

12.4.4 跨表间关联规则的提取

对于任意大项集 $l \in \text{multitemset}$, 若 u 为 l 的任意非空子集, 判断 $l-u \Rightarrow u$ 是否满足给定的支持度和置信度, 如是, 则输出对应的蕴含式以及支持度和置信度。另外, 对给定的大项集 $l \in \text{multitemset}$, 若 u 可作为蕴含式的后项, 则 u 的任意子集也可作为蕴含式的后项。反之, 若 u 的某一非空子集不能作为蕴含式的后项, u 也不可能成为蕴含式的后项, 故在生成蕴含式后项候选集 sc_i 时, 调用过程 `candi_gen()`。 u_j 表示 u 是 l 的一个含 j 个项的子集, 多关系间关联规则提取算法如下:

算法 12.5 多关系间关联规则提取算法。

```

forall large k-itemset l ∈ multitemset, k ≥ 2 do
    {forall large 1-itemset u ⊆ l do
        if l.sup/u1.sup ≥ minconf then
            {outputrule: attr_name(l-u1) ⇒ attr_name(u1)
              with support: l.sup and confidence: l.sup/u1.sup;
             insert u1 into s1;}
        for (i=2; si-1 ≠ ∅; i++) do
            {candi_gen(si-1, sci);
             forall ui ∈ sci do
                 if l.sup/ui.sup ≥ minconf then
                     {outputrule: attr_name(l-ui) ⇒ attr_name(ui)
                       with support: l.sup and confidence: l.sup/ui.sup;
                      insert ui into si;}
            }
    }

```

其中函数 `attr_name` 将属性编号转换为对应的属性名称。容易证明, 以上算法生成所有满足要求的多表间关联规则, 与先将多个表按相关对等属性自然连接后再行开采所得的结果一致。

综上所述, 数据挖掘可在任意多个表间同时进行, 并得到跨表间的蕴含规则。对于大数据集来说, 影响数据挖掘速度的主要因素是对数据集的访问方式和访问次数。对于顺序扫描数据集, 其扫描次数由可能形成的大项集中项的个数确定, 在最坏的情况下, 大项集中包含所有属性, 扫描次数等于数据集中属性的个数, 对于 $\delta_1, \delta_2, \dots, \delta_n$ 来说, 扫描次数的上界分别为 r_1, r_2, \dots, r_n , 因而是非常有效的。

12.5 基于分布式系统的关联规则挖掘算法

数据库或数据仓库可能存储相当大数量的数据，在这样的数据环境下进行关联规则的挖掘可能需要充足的处理器资源，而分布式系统是一个可能的解决方案。同时许多大型数据库本来就是分布式的。例如：超市发百货公司的数以万计的交易数据就很可能存在不同的地点，这种事实使得研究在数据库中挖掘关联规则的高效分布式算法显得非常必要，并同时带动并行算法的研究。因为分布式算法具有高度的适应性、可伸缩性、低性能损耗和容易连接等特性，它可以作为挖掘关联规则的理想平台。

由于有大量事务数据库的存在，这些数据库中存储着海量的数据，很容易想到将一个集中的数据库进行分割，从而利用分布式系统带来的高度的可伸缩性，达到提高效率的目的。D.W.Cheung 揭示了分散数据集与集中数据集之间的一些有趣关系，并提出了一个快速的基于分布式系统的关联规则挖掘算法 FDM，该算法通过生成数量较少的候选数据集，大大减少了在挖掘关联规则时需要处理的数据量

以事务数据库作为讨论对象，而相应的方法可以很容易地扩展到关系数据库中，该数据库中存储了大量的交易数据，每一个交易都有一个唯一的交易码（TID）和一组属性数据。此外，可以认为该数据库是“水平”分片的（例如：对交易进行分组），并且被分配在靠消息传递进行通讯的分布式系统中。

基于以上假设来考察对关联规则的分布式挖掘，挖掘关联规则的主要代价为对数据库中大数据集的计算。而对这些大数据集进行分布式计算会遇到一些新的问题。你可以在一个地方很容易地进行计算，但是一个局部的大数据集对于全局来说不一定是大数据集。因为对其他地点广播全部数据的代价是非常昂贵的，一种可行的做法是向其他地点广播数据集的聚合数据，而不考虑局部数据量的大小。但是，一个大数据库可能包括非常多数量的数据集的组合，这样，需要传输的信息量也是惊人的。

通过观察，可以发现在局部大数据集与全局大数据集之间，存在着一些有价值的关联。只要最大限度的利用这些关联，就可以减少信息的传输量，对需要局部处理的数据进行过滤。如前所述，目前已经存在两种挖掘关联规则的并行算法——PDM 和计数分布（CD）算法，它们都是基于各自独立的并行系统的，然而，它们也可以用在分布式环境中。FDM 相对于以上提出的两种算法，有着独特的特性：

1. 候选数据集的生成算法思想与 Apriori 算法类似。但是，在每个大数据量的重复数据集中生成小数据量的候选数据集的过程中，发现了一些关于局部的大数据集和全局的大数据集的有价值的关系。这样，就可以利用这些关系来减少信息传输量。
2. 在候选数据集被选出以后，在每一个单独的地点，都可以利用两种剪枝技术——局部剪枝和全局剪枝对候选数据集进行裁剪。
3. 为了决定一个候选集的数据量的大小，利用一个时间复杂度为 $O(n)$ 的算法来进行聚合数信息交换， n 代表整个网络的节点数。比起对 Apriori 算法进行直接的改编，其效率要高得多，因为后者的时间复杂度为 $O(n^2)$ 。

注意到在 FDM 算法中可以采用几种不同的局部剪枝和全局剪枝算法，着重研究了三个 FDM 的版本：FDM-LP，FDM-LUP，FDM-LPP，它们都具有相似的结构，但却具有不同的剪枝算法。FDM-LP 算法只讨论了局部剪枝；FDM-LUP 算法讨论了局部剪枝和上界剪枝；FDM-LPP 算法讨论了局部剪枝和逐点剪枝。

12.5.1 候选集的生成

在分布式环境中考察有关大数据集的某些特殊属性是非常重要的，因为这些属性可能被用来显著减少在挖掘关联规则时的网络信息传输量。

在大数据集与分布式数据库中的地点之间有一个重要的关系：每一个全局的大数据集必定在某一个地点是局部大数据集。如果一个数据集 X 在地点 S_i 既是全局大数据集又是局部大数据集。可以称 X 在地点 S_i 是全局大的，一个地点所有的全局大的数据集将作为该地点的候选数据集的源数据集。

可以观察到关于局部大数据集和全局大的数据集的两个特征：第一、如果一个数据集 X 在地点 S_i 是局部大的，那么它的所有子集在地点 S_i 也是局部大的。第二、如果一个数据集 X 在地点 S_i 是全局大的，那么它的所有子集在地点 S_i 也是全局大的。注意到在集中的环境中也有类似的关系，以下给出的是利用在分布式环境中有效生成候选集的技术得出的重要结果。

引理 12.1: 如果一个数据集 X 是全局大的，那么存在一个地点 S_i ($1 \leq i \leq n$)， X 以及它的所有子集在地点 S_i 是全局大的。

证明: 如果 X 在任何地点都不是局部大的，即 $X.\text{sup}_i < s * D_i$ ($i=1, \dots, n$)，于是， $X.\text{sup} < s * D_i$ ，即 X 不可能为全局大的，矛盾。所以 X 必定在某一个地点 S_i 是局部大的，故 X 在 S_i 是全局大的，从而 X 的所有子集在地点 S_i 是全局大的。□

用 GL_i 表示在地点 S_i 的全局大数据集， $GL_{i(k)}$ 表示在地点 S_i 的全局大的 k -数据集，根据引理 12.1，如果 $X \in L_{(k)}$ ，那么存在一个地点 S_i ($1 \leq i \leq n$)，使得 X 的所有大小为 $k-1$ 的子集在地点 S_i 是全局大的，也就是说，他们属于 $GL_{i(k-1)}$ 。

如果对 Apriori 算法进行直接的扩展，那么 Apriori 算法生成大小为 k 的结果数据集 $CA_{(k)}$ 表示经过 k 次迭代生成的候选集，也就是：

$$CA_{(k)} = \text{Arriori_gen}(L_{(k-1)})$$

在每一个地点，设 $CG_{i(k)}$ 为对 $GL_{i(k-1)}$ 施行 Apriori_gen 算法生成的候选数据集，也就是：

$$CG_{i(k)} = \text{Arriori_gen}(GL_{i(k-1)})$$

在这里， CG 代表从全局大的数据集中生成的候选集。因此 $CG_{i(k)}$ 由 $GL_{i(k-1)}$ 生成，而 $GL_{i(k-1)} \subseteq L_{(k-1)}$ ， $CG_{i(k)}$ 是 $CA_{(k)}$ 的子集。在以后的论述中，用 $CG_{(k)}$ 来代表集合 $\bigcup_{i=1}^n CG_{i(k)}$ 。

定理 12.1: 假定 $CG_{i(k)} = \text{Arriori_gen}(GL_{i(k-1)})$ ，对于每一个 $k > 1$ ，所有的全局大的 k -数据集 $L_{(k)}$ 是 $CG_{(k)} = \bigcup_{i=1}^n CG_{i(k)}$ 的子集。

证明: 设 $X \in L_{(k)}$ ，根据引理 12.1，存在一个地点 S_i ($1 \leq i \leq n$)， X 的所有大小为 $k-1$ 的子集在地点 S_i 是全局大的，因此， $X \in CG_{i(k)}$ 。所以，

$$L_{(k)} \subseteq CG_{(k)} = \bigcup_{i=1}^n CG_{i(k)} = \bigcup_{i=1}^n \text{Arriori_gen}(GL_{i(k-1)}).$$

定理 12.1 表明： $CG_{(k)}$ 是 $CA_{(k)}$ 的一个子集，并且有可能比 $CA_{(k)}$ 的数据量小很多，于是就可以作为大小为 k 的数据集的候选集。两个数据集 $CG_{(k)}$ 和 $CA_{(k)}$ 有多大差别取决于数据集的分布程度。这个定理奠定了 FDM 算法中生成候选数据集部分的基础。首先，候选数据集 $CG_{i(k)}$ 可以在每个地点 S_i 经过 k 次迭代在本地生成。经过数次支持集的交换，全局数据集 $GL_{i(k)}$ 就可以通过对 $CG_{i(k)}$ 迭代得到，在 $GL_{i(k)}$ 的基础上，又可以生成经过 $k+1$ 次迭代的候选数据集。根据第 5 节的性能研究，利用这种方式，候选数据集的数据量可以减少到 CD 算法

生成的数据量的 10-25%。

例 12.1 证明了利用定理一对减少数据量的有效性

例 12.1 假设某个系统中有三个分布地点将一个数据库系统 DB 分为 DB_1, DB_2, DB_3 。并假设大的 1-数据集（经过一层迭代计算所得） $L_{(1)} = \{A, B, C, D, E, F, G, H\}$ ，其中，A、B 和 C 在地点 S_1 是局部大的，B、C 和 D 在地点 S_2 是局部大的，E、F、G 和 H 在地点 S_3 是局部大的，所以， $GL_{1(1)} = \{A, B, C\}$ ， $GL_{2(1)} = \{B, C, D\}$ ， $GL_{3(1)} = \{E, F, G, H\}$ ，根据定理 12.1，在地点 S_1 的大小为 2 的候选数据集为 $CG_{1(2)}$ ， $CG_{1(2)} = \text{Apriori_gen}(GL_{1(2)}) = \{AB, BC, AC\}$ 。类似的， $CG_{2(2)} = \{BC, CD, BD\}$ ， $CG_{3(2)} = \{EF, EG, EH, FG, FH, GH\}$ ，因此，大的 2-数据集的候选数据集 $CG_2 = CG_{1(2)} \cup CG_{2(2)} \cup CG_{3(2)}$ ，共有 11 个候选元，但是，如果对 $L_{(1)}$ 直接进行 Apriori_gen 变换，那么候选数据集 $CA_{(2)} = \text{Apriori_gen}(L_1)$ 将包含 28 个元素。这说明利用定理 12.1 对减少候选数据集中的数据量是很有效的。

12.5.2 候选数据集的本地剪枝

根据定理 12.1，通常可以在分布式环境中选择生成一个比直接应用 Apriori 算法生成的数据集数据量小得多的候选数据集。

当候选数据集 $CG_{(k)}$ 生成成功后，为了得到全局大的数据集，就必须在所有地点之间交换支持元合计数的信息，注意到 $CG_{(k)}$ 中的某些候选数据集在进行合计数交换之前就可利用局部的剪枝技术进行剪枝。总的思想是：在每一个地点 S_i ，如果一个数据集 $X \in CG_{i(k)}$ 在地点 S_i 并不是局部大的，也就没有必要来算出它的全局大的支持元合计数来决定它是否是全局大的，这个结论是基于如下原因：如果 X 是小的（也就是说不是全局大的），或者它可能在别的地点是局部大的，那么，只有 X 为局部大的那些地点才有必要计算 X 的全局支持合计数。所以，为了计算所有的大的 k-数据集，在每一个地点 S_i ，候选数据集就可以只限定在数据集 $X \in CG_{i(k)}$ ，并且在地点 S_i 是局部大的。为了简略起见， $LL_{i(k)}$ 用来表示那些在 $CG_{i(k)}$ 中的候选集并且在地点 S_i 是局部大的。根据以上的讨论，在每一层迭代（共有 k 次迭代）的过程中，可以按照以下步骤计算出在地点 S_i 全局大的 k-数据集：

1. **候选集的生成：**根据在地点 S_i 经过 $k-1$ 次迭代生成的全局大的数据集的基础上，利用公式 $CG_{i(k)} = \text{Apriori_gen}(GL_{i(k-1)})$ 生成 $CG_{i(k)}$ 。
2. **本地剪枝：**对于每一个数据集 $X \in CG_{i(k)}$ ，扫描每一个局部数据库 DB_i 以计算本地支持合计数 $X.\text{sup}_i$ 。如果 X 在地点 S_i 不是局部大的，那么将其从候选数据集 $LL_{i(k)}$ 中删除。（注：这种剪枝只是将 X 从地点 S_i 的候选数据集中删除，X 还可能出现在别的地点的候选数据集中。
3. **支持合计数交换：**将 $LL_{i(k)}$ 中的候选元向其他地点广播，以收集支持合计数。计算全局的支持合计数，并得出在地点 S_i 所有全局大的 k-数据集。
4. **广播挖掘结果：**将计算所得的全局大的 k-数据集向其他地点广播。

为了便于理解，表 12.2 中列出了目前用到的所有符号。

表 12.2 符号表

D	DB 中的事务数
S	最小支持度 $minsup$
$L_{(k)}$	全局大的 k -数据集
$CA_{(k)}$	从 $L_{(k)}$ 中生成的候选数据集
$X.sup$	关于 X 的全局支持合计数
D_i	DB_i 中的事务数
$GL_{i(k)}$	在地点 S_i 全局大的 k -数据集
$CG_{i(k)}$	由 $GL_{i(k-1)}$ 生成的候选数据集
$LL_{i(k)}$	在 $CG_{i(k)}$ 中局部大的 k -数据集
$X.sup_i$	在地点 S_i 的局部支持合计数

为了举例说明以上步骤，继续对例 1 进行如下操作：

例 12.2：假设例 12.1 中的数据库包含 150 个事务，3 个局部数据库各包含 50 个事务，同时假设最小支持度 $s=10\%$ 。然后，根据例 12.1，经过第二次迭代，在地点 S_1 生成的候选数据集为 $CG_{1(2)}=\{AB, BC, AC\}$ ；在地点 S_2 生成的候选数据集为 $CG_{2(2)}=\{BC, AD, CD\}$ ；在地点 S_3 生成的候选数据集为 $CG_{3(2)}=\{EF, EG, EH, FG, FH, GH\}$ 。为了计算全局大的 2-数据集，必须首先计算局部的支持合计数，表 12.3 列出了所得计算结果。

表 12.3 局部大的数据集

$X.sup_1$		$X.sup_2$		$X.sup_3$	
AB	5	BC	10	EF	8
BC	10	CD	8	EG	3
AC	2	BD	4	EH	4
-	-	-	-	FG	3
-	-	-	-	FH	4
-	-	-	-	GH	6

从表 12.3 中可以看出， $AC.sup_1=2 < s \cdot D_1=5$ 。所以 AC 并不是局部大的，因此，候选元 AC 在地点 S_1 就在剪枝后被删除；另一方面，AB 和 BC 都有着足够大的局部支持合计数，因此，经过局部剪枝以后，它们仍然留在候选集中。所以， $LL_{1(2)}=\{AB, BC\}$ 。类似的， $LL_{2(2)}=\{BC, CD\}$ ， $LL_{3(2)}=\{EF, GH\}$ ，经过局部剪枝后，大小为 2 的候选集的数据量被减少到 5，这比原始数据量的一半还少。当局部剪枝完成后，每一个地点将留在候选数据集中的元素向其它地点广播，以便生成全局的支持合计数。表 12.4 中记录了支持合计数的交换情况。

算法需要将 AB 的支持合计数从地点 S_1 广播至地点 S_2 和地点 S_3 ，然后将传回的支持合计数存储在地点 S_1 ，如表 12.4 的第二行所示；在其它的行也记录下在其它的地点进行类似的支持合计数交换操作的结果，当本次迭代结束后，在地点 S_1 ，发现只有 BC 是全局大的，因为 $BC.sup = 22 > s \cdot D = 15$ ， $AB.sup = 13 < s \cdot D = 15$ 。因此，在地点 S_1 的全局大的 2-数据集为： $GL_{1(2)}=\{BC\}$ 。类似的， $GL_{2(2)}=\{BC, CD\}$ ； $GL_{3(2)}=\{EF\}$ 。经过向其它地点广播全局大的数据集以后，每一个地点都返回了全局大的 2-数据集 $L_{(2)}=\{BC, CD, EF\}$ 。

表 3.4 全局大的数据集

局部大的 候选元	广播 来源	X.sup ₁	X.sup ₂	X.sup ₃
AB	S ₁	5	4	4
BC	S ₁ , S ₂	10	10	2
CD	S ₂	4	8	4
EF	S ₃	4	3	8
GH	S ₃	4	4	6

注意到某些候选元，如本例中的 BC，可能在不止一个地点上是局部大的。在以上处理中，所有 BC 为局部大的地点都向其它地点广播有关 BC 的信息。这是不必要的，因为对每一个地点来说，相同的信息只要被广播一次就足够了，所以可以用一种优化技术以便消除这样的冗余。

在以上给出的寻找全局大的候选数据集的四个步骤的执行过程中，还有一个细微之处值得关注：为了支持步骤 2 “局部剪枝”和步骤 3 “支持合计数交换”，每个地点 S_i 都应有两个支持合计数集合。为了进行局部剪枝， S_i 必须基于它的本地候选集 $CG_{i(k)}$ 找出局部支持合计数；为了进行支持合计数交换， S_i 必须要找到那些与其它地点的全局支持合计数集合中元素不同的局部支持合计数。一个简单的步骤就需要对 DB_i 扫描两次，一次是为了找到基于它的本地候选集 $CG_{i(k)}$ 的局部支持合计数，另一次是为了响应其它地点对该地点支持合计数的请求，但是，这必定会明显地降低整个系统性能。

事实上，对数据库进行两次扫描并不是必要的，在地点 S_i ，在第 k 次迭代开始之前，不仅集合 $CG_{i(k)}$ 已经得到，一些其它的相关集合如 $CG_{j(k)}(j=1, \dots, n, j \neq i)$ 也可以得到，因为在 $k-1$ 次迭代结束后，所有的 $GL_{i(k-1)}(i=1, \dots, n)$ 都被广播至每一个地点，候选数据集 $CG_{i(k)}(i=1, \dots, n)$ 就可以从相应的 $GL_{i(k-1)}$ 中计算得出。也就是说，在每一次迭代的开始，因为前一次迭代产生的所有全局大的数据集都被广播至所有地点，每一个地点都可以计算出其它地点的候选数据集。那么，所有候选数据集的局部支持合计数都可以在一遍扫描中获得，并且可以被存储在一个类似于 Apriori 算法中的哈希树结构中，在局部剪枝和支持合计数交换中所需要的两套不同的支持合计数集合都可以在这个数据结构中检索得到。

12.5.3 候选数据集的全局剪枝

在地点 S_i 的局部剪枝中，只用到了在 DB_i 中得到的局部支持合计数对候选集进行剪枝，事实上，在其它地点得到的局部支持合计数也同样可以被用来剪枝，利用一种全局的剪枝技术来实施这样的剪枝，这种技术的要点如下：在每一次迭代结束时，可以得到候选数据集 X 的所有局部支持合计数和全局支持合计数，在一个候选数据集被确认为是全局大的以后，这些局部支持合计数和全局支持合计数都可以向所有地点进行广播，利用这一信息，就可以在以后的迭代中对候选数据集进行一些全局剪枝。

假设在每一次迭代结束后，如果发现某一候选数据集是全局大的，系统自动将每一个候选元的局部支持合计数向其它地点广播。假设 X 是一个在 k 次迭代结束后大小为 k 的候选数据集。因此，在每一个地点都已收到所有 X 的大小为 $k-1$ 的子集的局部支持合计数。对于一个

分支数据库 $DB_i (1 \leq i \leq n)$ 来说，用 $\maxsup_i(X)$ 来表示 X 的所有大小为 $k-1$ 的子集的最小的局部支持合计数，也就是说， $\maxsup_i(X) = \min \{Y.\sup_i | Y \subset X \text{ 且 } |Y|=k-1\}$ 。根据父集和子集之间的关系， $\maxsup_i(X)$ 就是局部支持合计数 $X.\sup_i$ 的上界函数。因此，所有分支数据库中这类上界函数的和，用 $\maxsup(X)$ 来表示，就是 $X.\sup$ 的上界。换句话说， $X.\sup \leq \maxsup(X) = \sum_{i=1}^n \maxsup_i(X)$ 。要注意的是： $\maxsup(X)$ 可以在每一次迭代的开始时在每一个地点计算出。因为 $\maxsup(X)$ 是它的全局支持合计数的上界，它就可以被用来进行全局剪枝。也就是说，如果 $\maxsup(X) < s * D$ ，那么 X 就不可能成为一个候选数据集。这种技术被称为全局剪枝。

利用全局剪枝和局部剪枝的不同组合可以形成不同的剪枝策略。在讨论 FDM 算法的不同版本时引入了两种不同的剪枝策略。第一种方法被称为上界剪枝，第二种方法被称为轮流检测剪枝。下面详细讨论上界剪枝，同时阐明轮流检测剪枝。在上界剪枝中，一个地点 S_i 首先生成候选数据集并对其进行局部剪枝。在合计数交换开始之前，地点 S_i 对余下的候选元进行全局剪枝。候选数据集 X 的一种可能的全局支持合计数上界为：

$$X.\sup_i + \sum_{j=1, j \neq i}^n \maxsup_j(X)$$

因为 $X.\sup_i$ 在局部剪枝后就可以获得，所以，该上界可以在地点 S_i 被计算出用以对候选数据集进行剪枝。

例 12.3：继续对例 2 中已经在 S_1 进行过局部剪枝后的候选数据集进行全局剪枝，根据表 12.3，经过局部剪枝后剩下的候选元为 AB 和 BC。在表 12.3 中可以找到它们在地地点 S_1 的局部支持合计数，更进一步，所有它们的子集在地地点 S_1 的局部支持合计数也可以得出，将其在表 12.5 中列出。

表 3.5 局部支持合计数

大的 1-数据集	在地地点 S_1 的局部支持合计数		
	$X.\sup_1$	$X.\sup_2$	$X.\sup_3$
A	6	4	4
B	10	10	5
C	4	12	5

根据表 12.3 和表 12.5，AB 的支持合计数的上界应该如下式，（用 $AB.\overline{\sup}$ 表示）

$$AB.\overline{\sup} = AB.\sup_1 + \min(A.\sup_2, B.\sup_2) + \min(A.\sup_3, B.\sup_3) = 5 + 4 + 4 = 13 < s * D.$$

因为该上界要小于最小支持度。所以应将 AB 从候选数据集中删去。

另一方面，BC 的支持合计数的上界应该如下式，（用 $BC.\overline{\sup}$ 表示）

$$BC.\overline{\sup} = BC.\sup_1 + \min(B.\sup_2, C.\sup_2) + \min(B.\sup_3, C.\sup_3) = 10 + 10 + 5 = 25 > s * D.$$

因为该上界要大于最小支持度。所以不应将 AB 从候选数据集中删去，并在地地点 S_1 仍然作为一个候选元。

全局剪枝对减少候选元素的个数方面是一个有效的技术，它的有效性取决于局部支持合计数的分布状况。

12.5.4 合计数轮流检测

在 CD 算法中, 每一个候选数据集的局部支持合计数被从一个地点向所有其它的地点进行广播, 如果设 n 为数据库分支数, 那么对于每一个候选数据集需要进行合计数交换的数量级为 $O(n^2)$ 。

如果一个候选数据集 X 在地点 S_i 是局部大的话, 那么 S_i 需要 $O(n)$ 数量级的信息来得到 X 的支持合计数, 通常来说, 在所有地点都是局部大的候选数据集时非常少的。所以, FDM 算法通常只需要少于 $O(n^2)$ 数量级的信息就可以计算出每一个候选元, 为了确保 FDM 在任何情况下只需要 $O(n)$ 数量级的信息就可以计算出每一个候选元, 可以引入一种合计数轮流检测的技术。

对于每一个候选数据集 X , 该技术用到了一个指派函数, 假设该函数为作用于 X 上的哈希函数, 将 X 映射为一个轮询地址 (假设该函数在任何一个地点都是可引用的。), 对应于 X 的轮询地址与 X 为局部大的那些地点是毫无关系的, 对于每一个候选数据集 X , 它的轮询地址是用来计算是否 X 为全局大的。为了达到这个目的, 对应于 X 的轮询地址必须向所有其它地点广播 X 的轮询请求, 收集局部支持合计数, 计算全局支持合计数。因为对应于每一个候选数据集 X , 有且仅有一个轮询地址, 所以 X 需要的合计数交换信息数就可以被减少到 $O(n)$ 数量级。

在进行 k 次迭代的过程中, 当剪枝阶段 (包括局部剪枝和全局剪枝) 结束后, FDM 在每一个地点 S_i 按照如下步骤进行合计数轮流检测。

- 1、将候选元送往各轮询地点: 假设在地点 S_i , 对于每一个轮询地址 S_j , 找到所有属于集合 $LL_{i,j(k)}$ 而且其轮询地址为 S_j 的候选元, 并将它们存储在集合 $LL_{i,j(k)}$ 中 (也就是说, 将候选元按照它们的轮询地址分组存放), 各候选数据集的局部支持合计数也同样存储在相应的集合 $LL_{i,j(k)}$ 中。然后将 $LL_{i,j(k)}$ 送往各自相应的轮询地址 S_j 。
- 2、轮流检测并收集支持合计数: 如果 S_i 是一个轮询地址, 那么 S_i 将接收所有来自其他地点的集合 $LL_{i,j(k)}$ 。对于每一个接收到的候选数据集 X , S_i 首先找到送出 X 的原始地点列表, 然后对每一个不再列表上的地点广播轮询请求, 以便收集支持合计数。
- 3、计算全局大的数据集: S_i 从别的地点接收支持合计数, 并为它的每一个候选元计算全局支持合计数, 然后找到全局大的数据集。最后, S_i 向其他地点广播它找到的全局大的数据集和它们的全局支持合计数。

例 12.4: 在例 12.2 中, 假设 S_1 被指定为 AB 和 BC 的轮询地址, S_2 被指定为 CD 的轮询地址, S_3 被指定为 EF 和 GH 的轮询地址。

根据以上指派, S_1 负责进行 AB 和 BC 的轮流检测, 以 AB 为例, S_1 将轮询请求送往 S_2 和 S_3 以便收集支持合计数, 对于 BC 来说, 因为它在 S_1 和 S_2 都是局部大的, 二元组 $\langle BC, BC.sup_2 \rangle = \langle BC, 10 \rangle$ 将被从 S_2 送往 S_1 。当 S_1 收到这个信息以后, 它将发送一个轮询请求给剩下的地点 S_3 , 一旦它收到来自 S_3 的支持合计数 $BC.sup_3=2$ 以后, S_1 计算出 $BC.sup=10+10+2=22>15$, 所以 BC 在地点 S_1 是全局大的。在本例中, 由于引入了一个轮询地址, 就避免了重复对各地点进行信息检测。

12.5.5 分布式挖掘关联规则的算法

在本节中, 首先给出 FDM 算法的基础版本, 即 FDM-LP (带局部剪枝的 FDM) 算法, 它包含了两项技术: 精简候选数据集和局部剪枝技术, 根据性能的分析, FDM-LP 的效率大大

优于 CD 算法。

算法 12.6 FDM-LP 算法：带局部剪枝的 FDM

输入：DB_i(i=1,...,n)，在地点 S_i 的分支数据库。

输出：L：所有全局大的数据集。

步骤：在每一个地点 S_i 分别反复执行以下程序片断(针对第 k 次迭代),一旦 L_(k)=∅或候选数据集 CG_(k)=∅，算法将终止。

1. if k = 1 then
2. T_{i(1)} = get_local_count(DB_i,∅,1)
3. else {
4. CG_(k) = $\bigcup_{i=1}^n$ CG_{i(k)}
 = $\bigcup_{i=1}^n$ Apriori_gen(GL_{i(k-1)});
5. T_{i(k)} = get_local_count(DB_i,CG_(k),i);}
6. For_all X ∈ T_{i(k)} do
7. If X.sup_i ≥ s*D_i then
8. for j = 1 to n do
9. if polling site(X) = S_j then
 insert <X,X.sup_i> into LL_{i,j(k)};
10. for j = 1,...,n do send LL_{i,j(k)} to site S_j;
11. for j = 1,...,n do {
12. receive LL_{j,i(k)} ;
13. for_all X ∈ LL_{j,i(k)} do {
14. if X ∉ LP_{i(k)} then
 insert X into LP_{i(k)} ;
15. update X.large_sites; }}
16. for_all X ∈ LP_{i(k)} do
17. send_polling_request(X);
18. reply_polling_request(T_{i(k)});
19. for_all X ∈ LP_{i(k)} do {
20. receive X.sup_j from the sites S_j,
 where S_j ∉ X.large_sites;
21. X.sup = $\sum_{i=1}^n$ X.sup_i ;
22. if X.sup ≥ s*D then
 insert X into G_{i(k)} ; }
23. broadcast G_{i(k)} ;
24. receive G_{j(k)} from all other sites S_j , (j ≠ i);
25. L (k) = $\bigcup_{i=1}^n$ G_{i(k)} .
26. divide L_(k) into GL_{i(k)} , (i = 1,...,n);
27. return L_(k).

在算法 FDM-LP 中，每一个 S_i 最开始是作为它所生成的候选数据集的“宿主地点”，然后成为了一个轮询地址来响应来自其他地点的请求，最后，它又被转换成了一个远程地点用以给其他轮询地址提供局部支持合计数。按照每一个 S_i 在算法 FDM-LP 中所起的不同作用的那些对应的行取出作专门讨论，如下所示：

1、宿主地点：生成候选数据集并将它们提交给其他地点（第 1-10 行）

在第一次迭代时，地点 S_i 调用函数 `get_local_count` 来扫描分支数据库 DB_i 一遍，并将所得到的基于所有的 1-数据集的局部支持合计数存储在数组 $T_{i(1)}$ 中。在进行 k ($k>1$) 次迭代时，地点 S_i 首先计算候选数据集 $CG_{(k)}$ ，然后构造一棵包含 $CG_{(k)}$ 中所有元素的局部支持合计数的哈希树 $T_{i(k)}$ 。通过遍历 $T_{i(k)}$ ， S_i 便可以找到所有局部大的 k -数据集，并将它们按照轮询地址分组，最终，它将候选数据集连同它们的局部支持合计数一并送往其对应的轮询地址。

2、轮询地址：接收候选数据集并发送轮询请求（第 11-17 行）

作为一个轮询地址，地点 S_i 从其他地点接收候选数据集并将它们插入到 $LP_{i(k)}$ 中，对于 $LP_{i(k)}$ 中的每一个候选元 X ， S_i 将它的“宿主”地址存储在 $X.large_sites$ 中，它包含所有将 X 送往 S_i 轮询的那些地点，为了对 X 进行合计数交换， S_i 调用函数 `send_polling_request` 来将 X 送往那些不在 $X.large_sites$ 中的地点以便收集余下的支持合计数。

3、远程地点：向轮询地点返回支持合计数（第 18 行）

当地点 S_i 从其他地点接收到轮询请求，它就已经作为一个远程地点了。对于每一个从其他地点接收到的候选集 Y ， S_i 从哈希树 $T_{i(k)}$ 中寻找 $Y.sup_i$ ，并将它返回给轮询地点。

4、轮询地址：接收支持合计数并找到大的数据集（第 19-23 行）

作为一个轮询地址， S_i 接收所有属于集合 $LP_{i(k)}$ 的候选数据集的局部支持合计数。然后，它再计算所有这些候选元的全局支持合计数并找出它们中间的全局大的数据集。这些全局大的 k -数据集将被存储在集合 $G_{i(k)}$ 中，最后， S_i 将集合 $G_{i(k)}$ 向所有其它地点广播。

5、宿主地点：接收大的数据集（第 24-27 行）

作为一个“宿主”地点， S_i 接收所有来自其他地点的全局大的 k -数据集 $G_{i(k)}$ ，经过对 $G_{i(k)}$ ($i=1, \dots, n$) 作并集运算， S_i 便可以得到所有大小为 k 的大数据集所组成的集合 L_k 。更进一步的操作是：利用 $X.large_sites$ 中的地点列表， S_i 便可以从 L_k 中得到对应于每一个地点的全局大数据集 $GL_{i(k)}$ ，而 $GL_{i(k)}$ 将在下一次迭代时用于候选数据集的生成。

在以下的讨论中，通过应用两种不同的全局剪枝技术，给出了 FDM-LP 算法的两种更精炼的版本。

算法 12.7 FDM-LUP：带有局部和上界剪枝的 FDM 算法。

方法：FDM-LUP 算法的主程序是在算法 FDM-LP 的第 7 行后加入如下条件（第 7.1 行）得到的。

7.1 if $g_upper_bound(X) \geq s * D$ then

FDM-LUP 算法的唯一一个更新是加入了上界剪枝（第 7.1 行）。函数 `g_upper_bound` 为每一个候选数据集 X 计算上界。换句话说，`g_upper_bound` 以如下公式给出 X 的上界：

$$X.sup_i + \sum_{j=1, j \neq i}^n \maxsup_j(X)$$

$X.sup_i$ 在局部剪枝阶段就已经计算过，而且 $\maxsup_j(X)$ ($j=1, \dots, n, j \neq i$) 的数值可以在 $k-1$ 次迭代结束后得到的局部支持合计数计算出。如果得出的上界要小于全局的最小支持度，那么它就可用来对 X 进行剪枝。FDM-LUP 算法与 FDM-LP 算法比较起来，通常得到的候选

元数量比较小。

算法 12.8 FDM-LPP: 带有局部和轮询地址剪枝的 FDM 算法。

方法: FDM-LUP 算法的主程序是将算法 FDM-LP 的第 17 行替换为如下两行得到的。

16.1 if $p_upper_bound(X) \geq s * D$ then

17 end_polling_request(X);

FDM-LPP 算法中加入的一个新步骤是轮询地址剪枝(第 16.1 行), 在该阶段, 是一个轮询地址, S_i 接收来自其它地点的请求, 并且进行轮询。每一个请求包含一个局部大的数据集 X 和它的局部支持合计数 $X.\sup_j$, S_j 是那些向 S_i 发送数据集 X 的地点, 注意到 $X.large_sites$ 是将包含 X 的轮询请求发往轮询地址的那些源地址的集合 (见第 15 行)。对于每一个地点 $S_j \in X.large_sites$, 它的局部支持合计数 $X.\sup_j$ 已经被送往地点 S_i 。对于每一个地点 $S_q \notin X.large_sites$, 因为 X 在 S_q 不是局部大的, 它的局部支持合计数 $X.\sup_q$ 必定小于局部最小支持度 $s * D_q$ 。 $X.\sup_q$ 应该小于或等于 $\min(\maxsup_q(X), s * D_q - 1)$ 。所以, $X.\sup_q$ 的上界可以由下式计算出:

$$\sum_{j \in X.large_sites} X.\sup_j + \sum_{q=1, q \notin X.large_sites}^n \min(\maxsup_q(X), s * D_q - 1)$$

在 FDM-LPP 算法中, S_i 调用 p_upper_bound 函数利用以上公式来为 $X.\sup$ 计算出一个上界。这个上界在小于全局支持合计数时被用来对 X 剪枝。

正如以前讨论的那样, FDM-LUP 和 FDM-LPP 算法都可能得到比 FDM-LP 算法小的候选数据集。但是, 它们需要对局部支持合计数准备更多的存储空间和交换信息。它们相对于 FDM-LP 算法的效率主要取决于数据的分布性。

12.6 词性标注规则的挖掘算法与应用

12.6.1 汉语词性标注

汉语词性自动标注问题是中文信息处理领域的基础性研究课题。它是进一步进行汉语句法分析的前提, 也可用于信息检索领域, 因此, 其研究具有非常重要的意义。从目前的研究上看, 基本上采用了基于概率统计和基于规则两种技术路线。其中, CLAWS 算法是基于概率统计的方法[Leech 94]。具体做法是: 首先对部分英文语料手工标注词性标记, 然后对标注好的语料进行统计, 得到标记与标记同现的频率, 最终产生一个同现概率矩阵。在词性标注时, 先取一个两端为非兼类词而中间为若干兼类词的片断 (SPAN)。在 SPAN 中, 词对应的词性标记的组合可以被视为多条路径。根据概率矩阵计算每条路径的概率, 并选择概率最大的路径上的词性标记作为兼类词的标记, 从而实现了兼类词的标注。

对于汉语词性标注问题采用了 CLAWS 算法的思想, 同时结合了每个词的各个词性标记具有不同概率的特点, 可以取得较好的标注效果。这种统计方法在训练语料规模足够大的情况下 (所要求的训练集规模应该与词性标记集的大小有关), 限于计算量成指数增长, 一般采用 bi-gram (二元语法, 即仅计算相邻标记的概率), 这使其正确率受到一定的影响。基于规则的方法通过考虑上下文中的词及标记对兼类词的影响决定兼类词的词性, 常常作为基于概率统计方法的补充。将统计方法和规则方法相结合被认为是解决词性标注问题的最佳手段[李晓黎

00]。

目前,规则的获得一般靠人工整理集成。但这存在以下两个方面的问题。一是从规则的应用范围上看,靠人工的方法只可能产生一些共性规则,不可能产生数量较多的针对个别情况的个性规则。而个性规则尽管应用范围小,但也是保证正确率的重要手段。二是人工方法产生规则的准确率有待验证。因此,在统计方法正确率不易再提高的前提下,能否自动高效地获取规则是实现汉语词性标注中的关键问题。

从人工已标注好的汉语文本语料中,利用文本挖掘来研究词及词性的模式序列对词性的影响是非常有新意的研究,这在国内外尚未见到有关报道。可以认为这与人在根据上下文对词性的判断方法是一致的,即不但根据上下文中的词性、词而且可根据二者的组合来判断某词的词性。在统计语料规模较大的情况下,给定最小支持度及最小可信度后,首先挖掘大于最小支持度常用模式集,然后生成关联规则。而若此规则的可信度大于最小可信度,则得到词性规则。只要最小可信度定义的足够高,获得的规则就可以用于处理兼类词的情况。这样获得的规则能够真正作为概率方法的补充,从而较好地解决汉语词性标注问题。但由于这种规则的条件依赖于词与词性的各种组合,同时又在文本数据中进行挖掘,这使得其挖掘过程比一般在数据库中的数据挖掘过程复杂得多。

12.6.2 问题的描述

为了在文本数据中挖掘词性标注规则,先给出如下的定义。

定义 12.8 词集 $DICT = \{WORD_i | i=1, 2, \dots, n\}$, 词性标记集 $TSET = \{TAG_i | i=1, 2, \dots, m\}$, 项集 $I = DICT \cup TSET$, 其中 $WORD_i$ 、 TAG_i 分别为某个汉语词或词性标记。

定义 12.9 已标记的文本 $T = \{(WORD_i, TAG_i) | WORD_i \in DICT, TAG_i \in TSET, TAG_i \text{ 是词 } WORD_i \text{ 在该标} \quad \text{记文本中对应的词性标记}\}$ 。例如: $\{(试析, VGN), (科技, NG), \dots\}$

定义 12.10 训练集 $E = \{(e_1, e_2, \dots, e_k) | e_i \in T, \text{ 且在文本中存在句子 } SENTENCE = e_1 e_2 \dots e_k, i=1, 2, \dots, K, K \in N\}$ 。这里 N 为自然数集合; 集合 E 中的每个元素为一个已标记的句子。例如: $\{((试析, VGN), (科技, NG), (情报, NG), (体制, NG), (改革, NG), (的, USDE), (热点, NG), (与, CW), (难点, NG)), ((本文, NG), (试图, VG), (运用, VGN), (统计, NG), (热力学, NG), (的, USDE), (观点, NG)), \dots\}$

定义 12.11 模式集 $D = \{d | d \in I^+\}$, 表示由词与词性标记组合构成的串。

定义 12.12 若 $X \in D$, 且其长度 $LEN(X) = K$, 则模式 X 为 K 模式。

定义 12.13 若 $X \in D, F = \{Y | Y \in D, \text{ 且 } LEN(X) = LEN(Y)\}$, 则 $X.SUP = \frac{freq(X)}{total(F)}$ 为

模式 X 的支持度, 它反映了该模式在同长度模式中所占的比例。其中, $freq(X)$ 表示模式 X 出现的频率; $Total(F)$ 表示长度为 $LEN(X)$ 的模式出现的总频率。

定义 12.14 令 $MINSUP$ 为用户规定的最小支持度。则集合 $C = \{X | X \in D, X.SUP \geq MINSUP\}$ 称为常用模式集。集合中的元素称为大模式。

定义 12.15 若 X, Y 为大模式, 则 X, Y 之间的关联, 记为规则 $X \Rightarrow Y$, 该规则的可信度

$(X \Rightarrow Y).CON$ 定义为 $PROB(Y/X) = \frac{freq(XY)}{freq(X)}$, 该规则的支持度定义为 $(X \cup Y).SUP$ 。

它表示了模式集 D 中, 包含 X 的模式中有多少同时包含了 Y , 其中 $\text{freq}(XY)$ 表示模式 X 、 Y 同现的频率。

定义 12.16 令 MINCON 为用户规定的最小可信度。若 $(X \Rightarrow Y) \cdot \text{CON} \geq \text{MINCON}$, 则规则 $X \Rightarrow Y$ 是值得该用户信赖的产生式规则。

定义 12.17 取 k 模式 $I_i = a_1 a_2 \cdots a_{k-1} a_k \in I$, 并且 $a_k \in \text{TSET}$, a_k 是词 k 的词性标记, 则采取的规则形式为: $\bigwedge_{j=1}^{k-1} a_j \Rightarrow (\text{词}k, a_k)$ 。它表明若前 $k-1$ 个词、标记构成的模式等于 $a_1 a_2 \cdots a_{k-1}$, 则第 k 个词 (词 k) 的词性标记应该为 a_k 。

定义 12.18 设 ε 是一个给定正数, 且 $\varepsilon > \text{MINSUP}$ 。

若规则 $X \Rightarrow Y$ 的支持度 $\text{MINSUP} < (X \cup Y) \cdot \text{SUP} < \varepsilon$, 则该规则为个性规则。

若规则 $X \Rightarrow Y$ 是个性规则, 且 $(X \Rightarrow Y) \cdot \text{CON} > \text{MINCON}$, 则该规则为个性可靠规则。

若规则 $X \Rightarrow Y$ 的支持度 $(X \cup Y) \cdot \text{SUP} > \varepsilon$, 则该规则为共性规则。

若规则 $X \Rightarrow Y$ 为共性规则, 且 $(X \Rightarrow Y) \cdot \text{CON} > \text{MINCON}$, 则该规则为共性可靠规则。

12.6.3 挖掘算法

词性标注规则的挖掘任务可以视为: 给定训练集 E , 根据 E 中的 WORD 和 TAG 两层结构对词的词性的影响, 求出所有满足最小支持度和最小可信度的产生式规则, 即个性可靠规则和共性可靠规则。

该问题可以分解为两个子问题: 一是求出 T 中满足最小支持度的常用模式集; 二是利用常用模式集生成满足最小可信度的关联规则。第二个问题的解决较为简单, 对每个常用模式, 可按照定义 12.17 的形式构造规则 (还可有其它形式, 但不在其中讨论)。如果 $(a_1 a_2 \cdots a_{k-1} \Rightarrow a_k) \cdot \text{CON} > \text{MINCON}$, 则将其加入规则集。关键技术是如何高效地求出常用模式集。

容易知道, 与数据库中的数据挖掘不同, 一个句子中长为 i 的子串中, 词与词性的组合数 (仅 i 模式) 为 2^i 。即随模式长度的增加, 该长度模式的总数 total 剧增, 所以最小支持度不应该是一成不变的。它们应随模式长度的增加而减少, 因而定义支持度向量为递减向量。但对最小可信度的要求不但不因模式长度的增加而减少, 而且由于长模式适用范围较小, 必须保证其可信度比短模式的可信度高, 否则将得不偿失, 因而, 可信度向量为递增向量。

由于挖掘工作是在文本数据中进行, 因此, 为了提高效率, 算法仅进行一遍扫描。

设最小支持度向量 $\text{minsup}[] = \{a_1 a_2 \cdots a_n\}$, 最小可信度向量 $\text{minconf}[] = \{b_1 b_2 \cdots b_n\}$, $\text{total}[i]$ 表示 i 模式总数 (初值为 0), $\text{MaxPatternSize} = W$ 表示用户规定的最长模式大小, pattern 为一个模式并且 pattern.count 存放该模式出现的次数, CandPatternSet 表示候选模式集 (初值为空集 $\{\}$), $\text{largePatternSet} = \{\}$ 表示最终形成的大模式集 (初值为空集 $\{\}$), 训练集为 $E = \{(e_1, e_2, \dots, e_k) \mid e_i \in T, i=1, 2, \dots, k\}$ 。则挖掘算法可描述为:

算法 12.9 词性标注规则的挖掘算法。

1. 若 $E = \emptyset$, 则转 STEP7。
2. 取 $(e_1, e_2, \dots, e_k) \in E$, 移动指针 $L=1$, $E \leftarrow E - \{(e_1, e_2, \dots, e_k)\}$ 。
3. 若 $K-L+1=0$, 则表示 (e_1, e_2, \dots, e_k) 已处理完毕, 转 STEP1。
4. 若 $K-L+1 > W$ 则取 $(\text{WORD}_L, \text{TAG}_L), \dots, (\text{WORD}_{L+W-1}, \text{TAG}_{L+W-1})$, $\text{LEN} \leftarrow W$,
否则取 $(\text{WORD}_L, \text{TAG}_L), \dots, (\text{WORD}_K, \text{TAG}_K)$, $\text{LEN} \leftarrow K-L+1$ 。
5. 循环 $j=2$ TO LEN 执行

- 构成包含 $(WORD_L, TAG_L)$ 的长度 j 的模式串 $pattern$,
 $pattern.count \leftarrow pattern.count + 1$,
 $CandPatternSet \leftarrow CandPatternSet \cup \{pattern\}$,
 $Total[j] \leftarrow total[j] + 1$.
6. $L \leftarrow L + 1$, 转 STEP3。
 7. 若 $CandPatternSet = \phi$, 则转 STEP11。
 8. 取 $pattern \in CandPatternSet$, $CandPatternSet \leftarrow CandPatternSet - \{pattern\}$ 。
 9. 若 $pattern.count / total[i] > minsup[i]$
 则 $LargePatternSet \leftarrow LargePatternSet \cup \{pattern\}$
 10. 转 STEP7。
 11. 若 $LargePatternSet = \phi$, 则算法结束。
 12. 取 $pattern = a_1 a_2 \cdots a_k \in LargePatternSet$,
 $LargePatternSet \leftarrow LargePatternSet - \{pattern\}$ 。
 13. 若 $a_k \in Tag$ and $(a_1 a_2 \cdots a_{k-1} \Rightarrow a_k).CON > MINCON[k]$, 则
 将规则 "if $a_1 a_2 \cdots a_{k-1} \Rightarrow (\text{词 } k, a_k)$ " 加入规则库,
 14. 转 STEP11。

算法的前 6 步是获取训练集 E 中每个例子的各长度的模式, 并放入候选模式集 $CandPatternSet$ 。7 至 10 步计算大模式集 $LargePatternSet$ 。11 步到 14 步从大模式集中产生规则并加入到规则库。

在规则库产生后, 先按规则长度从大到小的顺序排序, 以便在应用规则时利用较多的上下文信息, 提高词性排歧的准确率。对同长度的规则, 按支持度从大到小的顺序排列, 使共性规则排在前面, 从而提高规则的匹配效率。

为提高标注正确率, 将统计方法与规则方法相结合。对于兼类词的标注, 先看能否用规则处理。若能则直接标注词性, 从而利用了规则方法的高效性和高准确率。否则再利用概率统计方法标注。其标注过程如下述算法所示:

算法 12.10 统计方法与规则方法相结合算法。

1. 对待标注的文本 (已分词), 首先查非兼类词典和兼类词典标上所有可能的词性。
2. 取片断 SECTION (注意与 CLAWS 算法的 SPAN 不同)。它包括了一个兼类词序列以及序列前的若干非兼类词与序列后的一个非兼类词, 如下图所示:

Word ₁	Word ₂	Word ₃	Word _{i-1}	Word _i	Word _{i+1}	Word _j	Word _{j+1}
Tag ₁	Tag ₂	Tag ₃	Tag _{i-1}	Tag _{i,1}	Tag _{i+1,1}	Tag _{j,1}	Tag _{j+1}
					Tag _{i,2}	Tag _{i+1,2}	Tag _{j,2}	
					Tag _{i,3}			Tag _{j,3}	

其中 $Word_i$ 、 $Word_{i+1}$ 、.....、 $Word_j$ 为兼类词, 其余词为非兼类词。并且考虑到用户规定的最大模式长度为 w , 所以最多只取第一个兼类此前的 $w-1$ 个非兼类词 ($i \leq w$)。

3. 若不存在 SECTION, 则算法结束。
4. $L \leftarrow 1$ 。
5. 从 $(WORD_L, TAG_L)$ 、.....、 $(WORD_{i-1}, TAG_{i-1})$ 中构造模式, 并用之匹配规则。
 若匹配成功, 则按规则给 $Word_i$ 标注词性 $Tag_{i,1}^*$ 。此时认为 $Word_i$ 已是非兼类词。
 转 STEP2, 重新选取 SECTION。
6. 若 $L \leq i-1$ 则 $L \leftarrow L + 1$, 转 STEP5。
7. 取有两个非兼类词所夹将类词序列作为 SPAN, 如下图所示:

Word _{i-1}	Word _i	Word _{i+1}	Word _j	Word _{j+1}
Tag _{i-1}	Tag _{i,1}	Tag _{i+1,1}	Tag _{j,1}	Tag _{j+1}
	Tag _{i,2}	Tag _{i+1,2}	Tag _{j,2}	
	Tag _{i,3}			Tag _{j,3}	

计算由词性标记组成的各路径的概率。按最大路径中的标记给 Word_i Word_j 标注词性。

8. 转 2。

另外, 挖掘系统具有一定的开放性, 新语料加入后可随时进行累计挖掘, 以生成新的模式及规则。由于新语料加入后, 原来的规则可能因降低可信度和支持度而失效, 而一些原来因支持度或可信度低而被修剪的模式可能变成最终的规则, 因而, 必须用规则库的管理模块对规则进行有效的管理, (增删规则, 修改规则的可信度和支持度)。

12.6.4 试验结果

词性标记集以《“八五”汉语语料库的词性标记集》(1990 年 6 月第二稿) 与《汉语语料库语法标记》(1991 年 6 月第三稿) 为基础经部分调整而成。

在二十一万词的人工已标注词性语料的基础上, 将其划分为训练集和测试集。训练集包括二十万语料; 测试集含一万语料。

利用词性规则的挖掘算法, 分别对各长度的模式进行挖掘。从挖掘出的规则中, 可以看出词或词性及其组合对当前词的词性的影响。同时支持度表示规则的适用范围; 可信度表示规则的可靠程度。分别以各长度的模式进行说明。

一模式表示单个词或词性的出现次数。其中出现次数前十位的为: NG(34534)、VGN(8340)、VG(7898)、的(6900)、USDE(6742)、D(4934)、A(4378)、MX(3870)、RN(3260)、P(2992)。由于一模式中未利用上下文信息, 因而不构成规则。

二模式表示前一词或前一词性对当前词的词性的影响。可以注意到, 在所获得支持度大于 0.05 同时可信度大于 0.7 的 6659 条模式中, 有 6638 条说明前一词对当前词的词性的影响, 而剩余的 21 条是受前一词性的影响。仅可信度为 100% 的规则就有 5183 条(可信度小于 100% 的模式在括号中列出可信度), 下面仅举几例。

受前一词影响的如: 副 NG; 分之 MX; 炼成 NG; 即刻 VGO; 难于 VG; 久而久之 D; 乘机 VGO; 太 A; 晋绥 NG; 文职 NG; 袭占 NPL; 阻击 NG。

受前一词性影响的如: USSU VG、USSI NG、VNF NG、HM MX(0.974), NPFF NPFS(0.905)、VGA A(0.828)、HN NG(0.8), MF MX(0.727)、B NG(0.711)。

获取的标注规则为:

if (词 1, “副”) then (词 2, NG), 它表示若前一词为“副”, 则后一词的标记为名词。

三模式表示前二词或词性的组合对当前词的词性的影响。可以注意到, 在所获得支持度大于 0.005 同时可信度大于 0.7 的 81921 条模式中, 仅可信度为 100% 的规则就有 80647 条。下面仅举几例。如: 百 MF MX、的 固有 NG、MX QNI NPU、新华社 NPL T、A 物理 NG、个人不 VA、个体工商 NG、NPL 至 NPL、贷款 MX MW、NG 业 NOV、NG 越来越 A、上级主管 NG、NG 兼 NG、主任 NPFF NPFS、T 到 T、T 占领 NPL、S RN MG、QNI 本 NG、这 QNG NG(0.958)、RN 级 NG(0.952)、VGJ VGN NG(0.929)。

获取的标注规则为: if (词性 1, T) AND (词 2, “占领”) then (词 3, NPL)

限于篇幅, 四、五、六模式各举三例说明。四模式有: 可观 USDE 经济 NG、领取 营业 NG

FS、RN 已经 VGV VGN；五模式有：P 通过 试验 NG VG、FS NG USDE D FS、NG 以来 被 NG VGN；六模式有：RN 以 NPU 运河 VI、NG P NPL S 北 VGO、P NG 上 克服 了 NG。

经过对不同长度模式的比较可以清楚地看出：词在模式中的限制作用。在二模式中 NPFF NPFS 的可信度为 (0.905)，而在三模式中，主任 NPFF NPFS、政委 NPFF NPFS、管理员 NPFF NPFS 的可信度均为 100%。这显然是用了较多的上下文关系的缘故。注：NPFF 表示姓氏，NPFS 表示人名。

从试验数据上可以看出：随长度的增加，每种模式的组合可能性增多，模式的绝对数量增加。模式的支持度要减低。但由于受到较多的上下文的限制，一般可信度反而会增加。例如四模式共 394744 个可信度为一的有 275083 个。比例很高，即说明受到较多的制约后，词性被唯一确定的可能性增加了。

不难看出：模式中词对词性的影响更大，即词作为上下文因素之一，较词性具有更精确的限制作用（强限制）。这是因为，一个词性对应的词数远远多于一个词对应的词性个数。用词性做制约对应的上下文情况更多，不利于对兼类词进行排歧。但一般说来含词的模式的支持度要更小一些，获得的规则更倾向于个性规则。

从模式到规则可以有多种方式，只讨论由前面的词或词性来确定最后一词的词性的情况，即前制约的情况。上下文制约规则同样可以取得。

为了进行试验比较，先用纯统计方法对一万测试语料进行标注测试，准确性达 91.8%。之后，在获得标注规则的基础上，通过统计方法与规则方法的结合算法进行标注，正确率达到 96.2%。优于现有的词性标注系统。

根据汉语的特点，采用数据挖掘方法对汉语词性规则的自动获取进行了有益的尝试。这是一种从语料库中以规则的形式获取知识的新方式，较适用于大规模语料库的特点。在概率统计方法的基础上，将所获得的规则应用于词性标注，会提高标注的准确率。

附录：本节中用到的一些词性标记及含义

（参见“八五”汉语语料库的词性标记集，1990 年 6 月第二稿）

A 形容词 B 区别词 D 副词 FS 单纯方位词 HM 数词前缀
HN 名词前缀 MF 分数词 MG 概数词 MW 位数词 MX 系数词
P 介词 NG 普通名词 NPFF 姓氏 NPFS 人名 NPL 处所名词
NPU 机构和组织名 QNG 名量词 QNI 个体量词 S 处所词 T 时间词
USDE 结构助词“的” USSU 结构助词“所” USSI 结构助词“似的” VA 助动词
VG 一般动词 VGA 动词(带形容词宾语) VGN(带名词宾语) VGJ(带兼语宾语)
VGV(带动词宾语) VI 系动词

习 题

1. 举例说明什么是支持度、可信度、最小支持度、最小可信度和大项集。
2. 下表给出 4 个事务，最小支持度为 60%，请找出 1 项频繁项集。

TID	Date	Items_bought
1	01/05/2005	I1, I2, I4, I6
2	02/05/2005	I1, I2, I3, I4, I5
3	03/05/2005	I1, I2, I3, I5
4	04/05/2005	I1, I2, I4

3. 试述经典的 Apriori 算法。
4. 试述基于分布式系统的关联规则挖掘算法。
5. 什么是正相关，什么是负相关，举例说明强关联规则是负相关的情况。
6. 简述如何有效的挖掘如下规则，“一件免费商品可能触发在同一事务中 200 元的总购物”（约定每种商品的价格是非负的）。
7. 下表包括 9 个事务。假设最小支持度为 20%，请给出频繁项集。

表 事务项表

TID	Items
1	I1,I2,I5
2	I2,I4
3	I2,I3,I6
4	I1,I2,I4
5	I1,I3
6	I2,I3
7	I1,I3
8	I1,I2,I3,I5
9	I1,I2,I3

8. 如何进行增量式关联规则挖掘？

第十三章 知识发现

13.1 概述

知识发现是从数据集中抽取和精化新的模式。知识发现的范围非常广泛,可以是经济、工业、农业、军事、社会、商业、科学的数据或卫星观测得到的数据。数据的形态有数字、符号、图形、图象、声音等。数据组织方式也各不相同,可以是有结构、半结构、非结构的。知识发现的结果可以表示成各种形式,包括规则、法则、科学规律、方程或概念网。

目前,关系型数据库应用广泛,并且具有统一的组织结构,一体化的查询语言,关系之间及属性之间具有平等性等优点。因此,数据库知识发现(Knowledge Discovery in Databases, KDD)的研究非常活跃。该术语于1989年出现,Fayyad 定义为“KDD 是从数据集中识别出有效的、新颖的、潜在有用的,以及最终可理解的模式的非平凡过程” [Fayyad 1996a]。在上面的定义中,涉及几个需要进一步解释的概念:“数据集”、“模式”、“过程”、“有效性”、“新颖性”、“潜在有用性”和“最终可理解性”。数据集是一组事实 F (如关系数据库中的记录)。模式是一个用语言 L 来表示的一个表达式 E ,它可用来描述数据集 F 的某个子集 F_E , E 作为一个模式要求它比对数据子集 F_E 的枚举要简单(所用的描述信息量要少)。过程在 KDD 中通常指多阶段的一个过程,涉及数据准备、模式搜索、知识评价,以及反复的修改求精;该过程要求是非平凡的,意思是要有一定程度的智能性、自动性(仅仅给出所有数据的总和不能算作是一个发现过程)。有效性是指发现的模式对于新的数据仍保持有一定的可信度。新颖性要求发现的模式应该是新的。潜在有用性是指发现的知识将来有实际效用,如用于决策支持系统里可提高经济效益。最终可理解性要求发现的模式能被用户理解,目前它主要是体现在简洁性上。有效性、新颖性、潜在有用性和最终可理解性综合在一起可称之为兴趣性。

由于知识发现是一门受到来自各种不同领域的研究者关注的交叉性学科,因此导致了很多不同的术语名称。除了 KDD 称呼外,主要还有如下若干种称法:“数据挖掘”(data mining)、“知识抽取”(information extraction)、“信息发现”(information discovery)、“智能数据分析”(intelligent data analysis)、“探索式数据分析”(exploratory data analysis)、“信息收获”(information harvesting)和“数据考古”(data archeology)等等。其中,最常用的术语是“知识发现”和“数据挖掘”。相对来讲,数据挖掘主要流行于统计界(最早出现于统计文献中)、数据分析、数据库和管理信息系统界;而知识发现则主要流行于人工智能和机器学习界。

知识发现过程可粗略地理解为三部曲:数据准备(data preparation)、数据开采,以及结果的解释评估(interpretation and evaluation)(见图 13.1)。

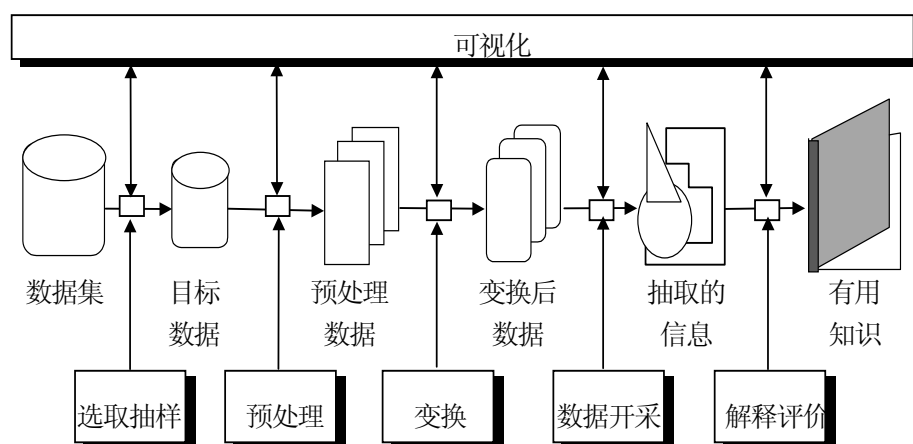


图13.1 知识发现过程示意图

13.1.1. 数据准备

数据准备又可分为三个子步骤：数据选取（data selection）、数据预处理（data preprocessing）和数据变换（data transformation）。数据选取的目的是确定发现任务的操作对象，即目标数据（target data），它是根据用户的需要从原始数据库中抽取的一组数据。数据预处理一般可能包括消除噪声、推导计算缺值数据、消除重复记录、完成数据类型转换（如把连续值数据转换为离散型的数据，以便于符号归纳，或是把离散型的转换为连续值型的，以便于神经网络归纳）等。当数据开采的对象是数据仓库时，一般来说，数据预处理已经在生成数据仓库时完成了。数据变换的主要目的是消减数据维数或降维（dimension reduction），即从初始特征中找出真正有用的特征以减少数据开采时要考虑的特征或变量个数。

13.1.2. 数据挖掘阶段

数据挖掘阶段首先要确定开采的任务或目的是什么，如数据总结、分类、聚类、关联规则发现或序列模式发现等。确定了开采任务后，就要决定使用什么样的开采算法。同样的任务可以用不同的算法来实现，选择实现算法有两个考虑因素：一是不同的数据有不同的特点，因此需要用与之相关的算法来开采；二是用户或实际运行系统的要求，有的用户可能希望获取描述型的（descriptive）、容易理解的知识（采用规则表示的开采方法显然要好于神经网络之类的方法），而有的用户或系统的目的是获取预测准确度尽可能高的预测型（predictive）知识。

完成了上述准备工作后，就可以实施数据挖掘操作了。具体的数据挖掘方法将在后面章节中作较为详细的论述。需要指出的是，尽管数据挖掘算法是知识发现的核心，也是目前研究人员主要努力的方向，但要获得好的采掘效果，必须对各种采掘算法的要求或前提假设有充分的理解。

13.1.3. 结果解释和评估

数据挖掘阶段发现出来的模式，经过用户或机器的评估，可能存在冗余或无关的模式，

这时需要将其剔除；也有可能模式不满足用户要求，这时则需要整个发现过程退回到发现阶段之前，如重新选取数据、采用新的数据变换方法、设定新的数据挖掘参数值，甚至换一种采掘算法（如发现任务是分类时，有多种分类方法，不同的方法对不同的数据有不同的效果）。另外，KDD 由于最终是面向人类用户的，因此可能要对发现的模式进行可视化，或者把结果转换为用户易懂的另一种表示，如把分类决策树转换为“if...then...”规则。

知识发现过程中要注意：

（1）数据挖掘仅仅是整个过程中的一个步骤。数据挖掘质量的好坏有两个影响要素：一是所采用的数据挖掘技术的有效性，二是用于采掘的数据的质量和数量（数据量的大小）。如果选择了错误的或不适当的属性，或对数据进行了不适当的转换，则采掘的结果不会好的。

（2）整个采掘过程是一个不断反馈的过程。比如，用户在采掘途中发现选择的数据不太好，或使用的采掘技术产生不了期望的结果；这时，用户需要重复先前的过程，甚至从头重新开始。

（3）可视化在数据挖掘的各个阶段都扮演着重要的作用。特别是，在数据准备阶段，用户可能要使用散点图、直方图等统计可视化技术来显示有关数据，以期对数据有一个初步的理解，从而为更好地选取数据打下基础。在开采阶段，用户则要使用与领域问题有关的可视化工具。在表示结果阶段，则可能要用到可视化技术。

13.2 知识发现的任务

13.2.1 数据总结

数据总结目的是对数据进行浓缩,给出它的紧凑描述。传统的也是最简单的数据总结方法是计算出数据库的各个字段上的求和值、平均值、方差值等统计值,或者用直方图、饼状图等图形方式表示。数据挖掘主要关心从数据泛化的角度来讨论数据总结。数据泛化是一种把数据库中的有关数据从低层次抽象到高层次上的过程。由于数据库上的数据或对象所包含的信息总是最原始、基本的信息(这是为了不遗漏任何可能有用的数据信息)。人们有时希望能从较高层次的视图上处理或浏览数据,因此需要对数据进行不同层次上的泛化以适应各种查询要求。数据泛化目前主要有两种技术:多维数据分析方法和面向属性的归纳方法。

多维数据分析方法是一种数据仓库技术,也称作联机分析处理(OLAP)。数据仓库是面向决策支持的、集成的、稳定的、不同时间的历史数据集合。决策的前提是数据分析。在数据分析中经常要用到诸如求和、总计、平均、最大、最小等汇集操作,这类操作的计算量特别大。因此一种很自然的想法是,把汇集操作结果预先计算并存储起来,以便于决策支持系统使用。存储汇集操作结果的地方称作多维数据库。多维数据分析技术已经在决策支持系统中获得了成功的应用,如著名的 SAS 数据分析软件包、Business Object 公司的决策支持系统 Business Object,以及 IBM 公司的决策分析工具都使用了多维数据分析技术。

采用多维数据分析方法进行数据总结,它针对的是数据仓库,数据仓库存储的是脱机的历史数据。为了处理联机数据,研究人员提出了一种面向属性的归纳方法。它的思路是,直接对用户感兴趣的数据视图(用一般的 SQL 查询语言即可获得)进行泛化,而不是像多维数据分析方法那样预先就存储好了泛化数据。方法的提出者对这种数据泛化技术称之为面向属性的归纳方法。原始关系经过泛化操作后得到的是一个泛化关系,它从较高的层次上总结了在低层次上的原始关系。有了泛化关系后,就可以对它进行各种深入的操作而生成满足用户需要的知识,如在泛化关系基础上生成特性规则、判别规则、分类规则,以及关联规则等。

13.2.2 概念描述

用户常常需要抽象的有意义的描述。经过归纳的抽象描述能概括大量的关于类的信息。有两种典型的描述：特征描述和判别描述。从与学习任务相关的一组数据中提取出关于这些数据的特征式。这些特征式表达了该数据集的总体特征。而判别描述则描述了两个或更多个类之间有何差异。

13.2.3 分类

分类在数据挖掘中是一项非常重要的任务,目前在商业上应用最多。分类的目的是学会一个分类函数或分类模型(也常常称作分类器),该模型能把数据库中的数据项映射到给定类别中的某一个。分类和回归都可用于预测。预测的目的是从利用历史数据纪录中自动推导出对给定数据的推广描述,从而能对未来数据进行预测。和回归方法不同的是,分类的输出是离散的类别值,而回归的输出则是连续数值。这里我们将不讨论回归方法。

要构造分类器,需要有一个训练样本数据集作为输入。训练集由一组数据库记录或元组构成,每个元组是一个由有关字段(又称属性或特征)值组成的特征向量,此外,训练样本还有一个类别标记。一个具体样本的形式可为: $(v_1, v_2, \dots, v_n; c)$; 其中 v_i 表示字段值, c 表示类别。

分类器的构造方法有统计方法、机器学习方法、神经网络方法等等。统计方法包括贝叶斯法和非参数法(近邻学习或基于范例的学习),对应的知识表示则为判别函数和原型事例。机器学习方法包括决策树法和规则归纳法,前者对应的表示为决策树或判别树,后者则一般为产生式规则。神经网络方法主要是 BP 算法,它的模型表示是前向反馈神经网络模型(由代表神经元的节点和代表联接权值的边组成的一种体系结构),BP 算法本质上是一种非线性判别函数。另外,最近又兴起了一种新的方法:粗糙集(rough set),其知识表示是产生式规则。

不同的分类器有不同的特点。有三种分类器评价或比较尺度:1 预测准确度;2 计算复杂度;3 模型描述的简洁度。预测准确度是用得最多的一种比较尺度,特别是对于预测型分类任务,目前公认的方法是 10 番分层交叉验证法。计算复杂度依赖于具体的实现细节和硬件环境,在数据挖掘中,由于操作对象是巨量的数据库,因此空间和时间的复杂度问题将是非常重要的一个环节。对于描述型的分类任务,模型描述越简洁越受欢迎;例如,采用规则表示的分类器构造法就更有用,而神经网络方法产生的结果就难以理解。

另外要注意的是,分类的效果一般和数据的特点有关,有的数据噪声大,有的有缺值,有的分布稀疏,有的字段或属性间相关性强,有的属性是离散的而有的是连续值或混合式的。目前普遍认为不存在某种方法能适合于各种特点的数据。

13.2.4 聚类

根据数据的不同特征,将其划分为不同的数据类。它的目的是使得属于同一类别的个体之间的距离尽可能的小,而不同类别上的个体间的距离尽可能的大。聚类方法包括统计方法、机器学习方法、神经网络方法和面向数据库的方法。

在统计方法中,聚类称聚类分析,它是多元数据分析的三大方法之一(其它两种是回归分析和判别分析)。它主要研究基于几何距离的聚类,如欧式距离、明考斯基距离等。传统的统计聚类分析方法包括系统聚类法、分解法、加入法、动态聚类法、有序样品聚类、有重叠聚类和

模糊聚类等。这种聚类方法是一种基于全局比较的聚类,它需要考察所有的个体才能决定类的划分;因此它要求所有的数据必须预先给定,而不能动态增加新的数据对象。聚类分析方法不具有线性的计算复杂度,难以适用于数据库非常大的情况。

在机器学习中聚类称作无监督或无教师归纳;因为和分类学习相比,分类学习的例子或数据对象有类别标记,而要聚类的例子则没有标记,需要由聚类学习算法来自动确定。很多人工智能文献中,聚类也称概念聚类;因为这里的距离不再是统计方法中的几何距离,而是根据概念的描述来确定的。当聚类对象可以动态增加时,概念聚类则称是概念形成。

在神经网络中,有一类无监督学习方法:自组织神经网络方法;如 Kohonen 自组织特征映射网络、竞争学习网络等等。在数据挖掘领域里,见报道的神经网络聚类方法主要是自组织特征映射方法,IBM 在其发布的数据挖掘白皮书中就特别提到了使用此方法进行数据库聚类分割。

13.2.5 相关性分析

发现特征之间或数据之间的相互依赖关系。数据相关性关系代表一类重要的可发现的知识。一个依赖关系存在于两个元素之间。如果从一个元素 A 的值可以推出另一个元素 B 的值 ($A \rightarrow B$), 则称 B 依赖于 A。这里所谓元素可以是字段,也可以是字段间的关系。数据依赖关系有广泛的应用。

依赖关系分析的结果有时可以直接提供给终端用户。然而,通常强的依赖关系反映的是固有的领域结构而不是什么新的或有趣的事物。自动地查找依赖关系可能是一种有用的方法。这类知识可被其它模式抽取算法使用。常用技术有回归分析、关联规则、信念网络等。

13.2.6 偏差分析

如分类中的反常实例、例外模式、观测结果对期望值的偏离以及量值随时间的变化等。其基本思想是寻找观察结果与参照量之间的有意义的差别。通过发现异常,可以引起人们对特殊情况的加倍注意。异常包括如下几种可能引起人们兴趣的模式:不满足常规类的异常例子;出现在其它模式边缘的奇异点;与父类或兄弟类不同的类;在不同时刻发生了显著变化的某个元素或集合;观察值与模型推测出的期望值之间有显著差异的事例等。偏差分析的一个重要特征就是它可以有效地过滤大量的不感兴趣的模式。

13.2.7 建模

通过数据挖掘,构造描述一种活动或状态的数学模型。机器学习中的知识发现,实际上就是对一些自然现象进行建模,重新发现科学定律,如 BACON[Shi 1992]。Langley, Simon, 和 Bradshaw 等于 1976 到 1983 年研制了六个版本的 BACON 系统。它们具有如下共同特点:采用数据驱动,通过启发式约束搜索,依赖于理论数据项,递归应用一些通用的发现方法。程序反复检查数据,并用改进操作来产生新的“项”,直到发现一个项总是常量,这时就可以把概念表示为“项=常量”的形式。

13.3 知识发现工具

随着大规模的数据库迅速地增长,人们对数据库的应用已不满足于仅对数据库进行查询和检索。仅用查询检索不能提取数据中蕴藏的丰富知识,将信息变为知识,必须通过数据挖掘,从海量数据中发现富有意义的知识。另一方面,从人工智能来看,专家系统的研究虽然取得了一定的进展。但是,知识获取仍然是专家系统研究中的瓶颈。知识工程师从领域专家处获取知识是非常复杂的个人到个人之间的交互过程。具有很强的个性和随机性,没有统一的办法。因此,人们开始考虑以数据库作为新的知识源。知识发现能自动地处理数据库中大量的原始数据,抽取出具有必然性的、富有意义的模式,作为有助于人们实现其目标的知识,找出人们对所需问题的解答。

这里列出一些具有代表性的知识发现工具或平台:

1. SAS Enterprise Miner

美国 SAS 公司的 SAS Enterprise Miner 是一种通用的数据挖掘工具,以其强大的数据管理能力、全面的统计方法、高精度的计算以及独特的多平台自适应技术,使其成为统计软件包的标准,被国内外许多学者誉为最权威的优秀统计软件包。在 80 年代进入中国后,占据了许多大型部门的统计室。目前 SAS 对 Windows 和 Unix 两种平台都提供支持。SAS 提供“数据步”和“过程步”两种处理数据的方式,可进行复杂而灵活的统计分析。通过收集分析各种统计资料和客户购买模式,SAS Enterprise Miner 可以帮助您发现业务的趋势,解释已知的事实,预测未来的结果,并识别出完成任务所需的关键因素,以实现增加收入、降低成本。

SAS Enterprise Miner 提供“抽样-探索-转换-建模-评估”(SEMMA)的处理流程。数据挖掘算法有:

- 聚类分析, SOM/KOHONEN 神经网络分类算法
- 关联模式/序列模式分析
- 多元回归模型
- 决策树模型(C45, CHAID, CART)
- 神经网络模型(MLP, RBF)
- SAS/STAT, SAS/ETS 等模块提供的统计分析模型和时间序列分析模型也可嵌入其中。

2. Intelligent Miner

IBM 公司的 Intelligent Miner 具有典型数据集自动生成、关联发现、序列规律发现、概念性分类和可视化显示等功能。它可以自动实现数据选择、数据转换、数据发掘和结果显示。若有必要,对结果数据集还可以重复这一过程,直至得到满意结果为止。

IBM 的 Intelligent Miner 已形成系列,它帮助用户从企业数据资产中识别和提炼有价值的信息。它包括分析软件工具 ----Intelligent Miner for Data 和 IBM Intelligent Miner for Text。Intelligent Miner for Data 可以寻找包含于传统文件、数据库、数据仓库和数据中心中的隐含信息。IBM Intelligent Miner for Text 允许企业从文本信息中获取有价值的客户信息。文本数据源可以是 Web 页面、在线服务、传真、电子邮件、Lotus Notes 数据库、协定和专利库。

3. Clementine

Solution 公司的 Clementine 提供了一个可视化的快速建立模型的环境。它由数据获取 (Data Access)、探查 (Investigate)、整理 (Manipulation)、建模 (Modeling) 和报告 (Reporting) 等部分组成。都使用一些有效、易用的按钮表示, 用户只需用鼠标将这些组件连接起来建立一个数据流, 可视化的界面使得数据挖掘更加直观交互, 从而可以将用户的商业知识在每一步中更好的利用。该系统具有下列特色:

- 可以用图分析探测数据: 直方图、分布图可以清楚展现数据的内部结构, 并且可以从图形中直接生成新的变量或者对数据进行平衡处理; 散点图、网状图可以检测不同变量间的关系的强弱, 形象地加以刻画, 并生成新的变量或者对数据进行筛选;

- 可以从多种建模技术中选择合适的模型:

- ①规则归纳模型: C5.0 和 BuildRule 可以生成决策树, 生成易懂的模型, 并进行预测; GRI 和 Apriori 可以自动检测出复杂的关系, 建立预测模型。

- ②神经网络模型: MLP、RBFN 和 Kohonen 网络可以从训练数据中进行自学习, 解读复杂的关系, 建立预测模型, Kohonen 网络可以将数据按照相似程度加以分类, 比如客户、订单。

- ③K-means 是一种快速高效的聚类算法;

- ④回归可以用于预测等。

- 可以将多种模型技术组合起来或者建立大型模型 (meta-models)
- 将 SPSS 集成在 Clementine 中
- 可以直接读写 SPSS 数据文件, 可以使用 SPSS 进行数据准备、报告、深度数据分析、作图等, 可以调用 SPSS 所有分析方法, 可以在 SPSS 或者 Clementine 中显示结果
- 开放型的数据挖掘平台, 可以通过外部模块接口添加更多的算法, 可以以批处理的模式运行。

4. Cognos Scenario

加拿大 Cognos 公司创立于 1969 年, 总部设在首都渥太华, 推出一系列商务智能软件。Cognos Scenario 是基于树的高度视图化的数据挖掘工具, 它将信息挖掘自动化。决策树的基本功能是创立一系列标准, 预测记录中目标市场的价值。Scenario 的分类树分阶展现各种因素; 最终用户通过挖掘或展开树的分支来探索数据。Scenario 可以帮助企业经理指导分析, 并用一套严格的标准测量信息。由于有时数据集庞大而笨重, Scenario 的抽样技术可以用最少的处理开销和最短的响应时间得出最精确的结果。为与 Cognos 的“分析 然后 查询”和“查询 然后 导航”的方法相一致, Scenario 与 PowerPlay 和 Impromptu 相集成, 允许用户发现分析查询。

人们可以利用 Scenario 的统计方法, 深入挖掘影响商务趋势的因素的潜在含义, 根据风险特性将个体与群体客户归类; 将商务因素分门别类, 辨清商务目标所受的主要影响; 探索与通常数据模式不符的异常情况。

5. MSMiner

中科院计算技术研究所智能信息处理开放实验室开发的 MSMiner 是一种多策略知识发现平台,能够提供快捷有效的数据挖掘解决方案,提供多种知识发现方法。MSMiner 具有下列特点:

- 提出了一种面向对象的元数据结构,将经过良好封装的元数据对象以层次结构组织起来,形成一种元数据对象模型,并通过这种元数据对象模型统一存取和管理元数据,使系统具有良好的一致性和可维护性。

- 设计实现了一种简单但有效的数据仓库平台,按主题组织数据,以星型模式建模,提供了有效的数据抽取和集成功能,为数据挖掘任务提供经过良好处理的数据来源。

- 提出了一种面向对象的数据挖掘任务模型。数据挖掘任务的每个步骤都用对象来表示,每个对象包含多种属性以及 DML 方法脚本,各个步骤对象通过有向图模型组织起来。通过这种任务模型能够有效表达各种数据挖掘任务。MSMiner 系统实现了可视化的任务编辑环境,以及功能强大的任务处理引擎,能够快捷有效地实现各种数据挖掘任务。

- 设计了一种可扩展算法库,以动态链接库 DLL 的方式集成了各种数据挖掘算法,并设计了开放的接口,能够灵活扩展用户自定义算法。

一些研究单位提供了知识发现公用系统,例如斯坦福大学的 MLC++、华盛顿大学的 Brute 等,读者可以通过 Internet 网免费下载。

美国 MathWorks 公司于 1982 年推出的一套高性能的数值计算和可视化数学软件,被称为 MATLAB。MATLAB 的含义是矩阵实验室(MATRIX LABORATORY),主要用于方便矩阵的存取,其基本元素是无须定义维数的矩阵。它集数值分析、矩阵运算、信号处理和图形显示于一体,构成了一个方便的、界面友好的用户环境。在这个环境下,对所要求解的问题,用户只需简单地列出数学表达式,其结果便以数值或图形方式显示出来。MATLAB 中包括了被称作工具箱(TOOLBOX)的各类应用问题的求解工具。工具箱实际上是对 MATLAB 进行扩展应用的一系列 MATLAB 函数(称为 M 文件),它可用来求解各类学科的问题,包括信号处理、图象处理、控制系统辨识、神经网络等。随着 MATLAB 版本的不断升级,其所含的工具箱的功能也越来越丰富,因此,应用范围也越来越广泛,成为涉及数值分析的各类工程师不可不用的工具。MATLAB 中的模块也可以用来快速开发知识发现工具。

下面我们将以知识发现平台 MSMiner 为例[Shi 2000],介绍知识发现的关键技术。

13.4 MSMiner 的体系结构

中科院计算技术研究所智能信息处理开放实验室开发的知识发现平台 MSMiner 采用任务驱动模型组织挖掘过程,元数据作为系统的管理和调度中心,实现了数据仓库与数据挖掘的有机集成和数据挖掘算法与采掘任务的无缝连接。

13.4.1 数据挖掘模型

我们为 MSMiner 定义了包含三个逻辑层次的数据挖掘系统模型,如图 13.2 所示。

决策分析需要大量经过良好组织和综合的数据,数据挖掘尤其依赖于经过一定预处理的

清洁的数据，这些数据来源于外部各种数据源，需要经过复杂的抽取和整合，集成到数据仓库中去，这些工作都由数据获取层来完成。因此数据获取层在整个系统中占有非常重要的地位。

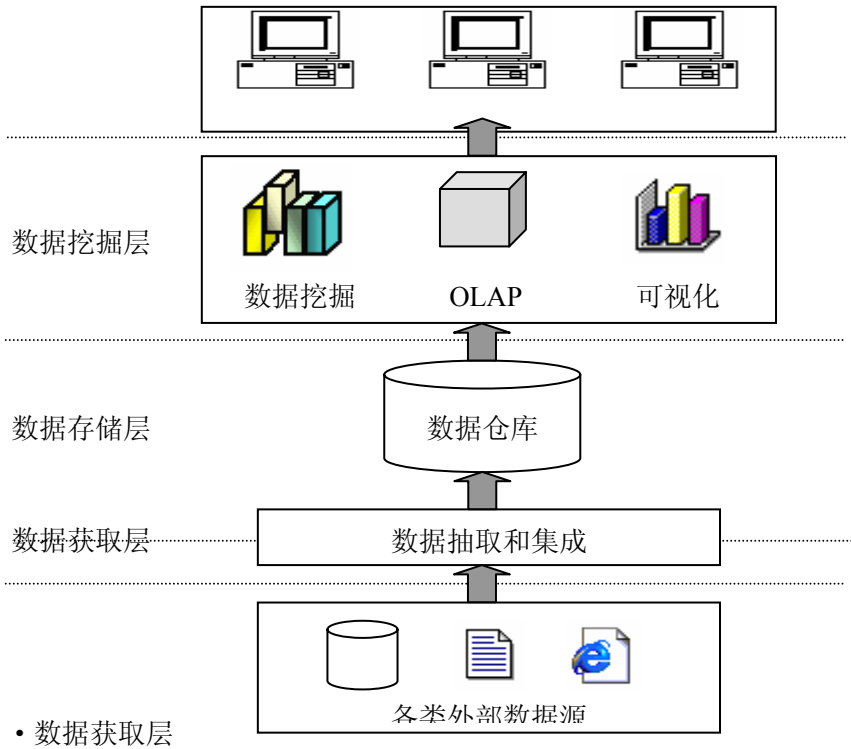


图 13.2 MSMiner 数据挖掘模型

MSMiner 的数据获取层主要包括一个数据抽取模块，通过这个模块可从各种关系型数据库、数据文件以及 Web 数据等外部数据源中抽取合适的数据，并进行各种清洗、整合和转换处理，将数据集成到数据仓库中的各个主题中去。

• 数据存储层

数据存储层就是数据仓库本身，包括一个或多个数据库，以一定的组织结构存储各种集成的数据。

MSMiner 的数据存储层以多个 SQL Server 数据库实现。MSMiner 数据仓库包含多个主题，每个主题存储在一个数据库中，包括多个综合表，这些综合表主要有以下三种：事实表、维表和为数据挖掘生成的中间表。

• 数据挖掘层

数据挖掘层是数据挖掘系统模型中的最高层，也往往是用户最关心的一层。数据仓库中的大量集成数据要为用户所用，所用的基于数据仓库的数据分析和数据挖掘功能就在这一层通过集成各种分析和采掘工具来实现。

MSMiner 的数据挖掘层是系统功能实现的重点，主要包括一个功能强大的数据挖掘集成工具，集成了各种数据挖掘算法，并提供灵活有效的任务模型组织形式，可以支持各种数据挖掘任务。

MSMiner 的数据挖掘层还包括一个 OLAP 工具，通过建立数据立方体实现各种多维数据分析处理操作；以及各种集成的可视化工具，以多种方式直观有效地表示数据和数据挖掘以及 OLAP 操作的结果。

13.4.2 系统功能

MSMiner 致力于提供一个功能强大的数据分析和数据挖掘集成环境。包括一个数据仓库平台，提供基于数据仓库的数据管理和面向决策的数据分析处理功能；和一个建立在数据仓库基础之上的数据挖掘工具，提供各种数据挖掘算法以及灵活开放的任务组织形式，能够有效完成各种数据挖掘任务。

MSMiner 的最终设计目标旨在为各种科学研究和决策支持应用提供快捷有效的数据挖掘解决方案。MSMiner 提供的功能包括：

- 从多种操作数据源抽取数据的能力、以及跨数据源的数据集成能力；
- 集中管理和维护数据仓库中数据的功能，包括数据存储优化、数据增量维护的能力；
- 集成 OLAP 多维综合和分析，通过内置 OLAP 引擎，提供高效 OLAP 分析的能力；
- 集成各种数据挖掘算法，通过灵活的任务模型组织方式，提供处理各种数据挖掘任务的能力；并提供开放的接口，提供扩展用户自定义算法的能力。
- 提供多种可视化方法显示和分析各种数据和数据挖掘结果的能力。

MSMiner 提供对数据的开放性，这对任何一个数据仓库的解决方案都是十分关键的。企业的数据库系统可能是多种多样的，MSMiner 的数据抽取引擎提供访问其它任意支持 ODBC（开放数据库连接）的数据库的能力。系统为数据挖掘任务提供了可扩展的数据挖掘算法库，用户可以根据自己的需要，通过系统提供的开放接口，加入自定义的算法，使系统适用于各种不同的数据挖掘任务。在提供向外部数据源的开放性的同时，MSMiner 特别注意了元数据的设计、数据仓库的建模、OLAP 引擎的集成以及数据挖掘任务处理引擎的设计，以达到最优化的性能。MSMiner 提供便捷有效的数据挖掘实现方案，因此必须为用户提供简便易行的处理各种数据挖掘任务的手段。MSMiner 系统以向导方式实现各个功能模块，并提供了各种可视化处理界面，使用户能够直观便捷地描述和完成各种数据分析和数据挖掘任务。

13.4.3 体系结构

为实现上述设计目标，MSMiner 采用客户 / 服务器方式构建。服务器端为实现数据仓库的 SQL Server 7.0 数据库服务器。客户端为 MSMiner 前端系统，主要包括三个部分：数据仓库管理器、数据挖掘集成工具和面向对象的元数据模块。整个系统的结构如图 13.3 所示。

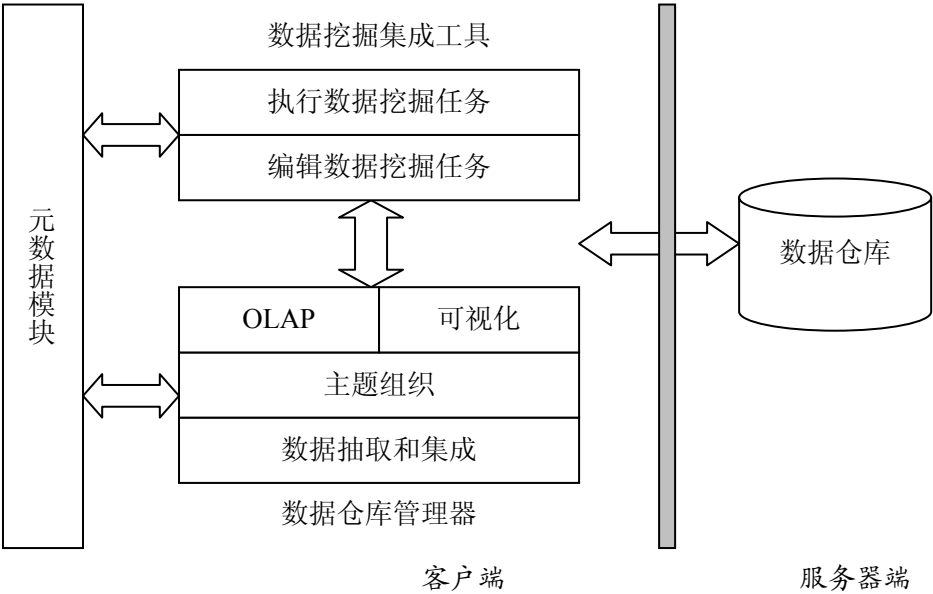


图 13.3 MSMiner 体系结构示意图

数据仓库管理器包含了数据仓库各种主要功能的实现，包括数据抽取和集成、数据仓库主题的组织、OLAP 和各种可视化功能的实现等。数据仓库管理器负责管理数据仓库中的数据，为整个系统提供数据平台，为各种数据分析处理以及更高层的数据挖掘提供统一的数据环境。

数据挖掘集成工具建立在数据仓库平台之上，完成数据挖掘任务的组织、任务的规划和解释执行以及结果的解释和评价等数据挖掘高级功能。

数据仓库管理器为数据挖掘集成工具提供经过清洗和整合的数据，是整个系统的数据平台；数据挖掘集成工具为数据仓库提供高层次的数据分析处理和信息采掘功能，是系统的实现重点。

MSMiner 系统的各个部分都由元数据统一管理和监控。从数据的抽取和管理到数据挖掘任务的建立和执行，整个流程都在元数据的控制之下。元数据在 MSMiner 系统中居于核心地位，是整个系统的灵魂和中枢。系统采用面向对象的方式，建立了一种元数据对象模型，通过各种元数据对象实现对元数据的存取和管理，以保证系统的一致性和可操作性。

小结：本节介绍了 MSMiner 的体系结构。我们认为，将 MSMiner 定位于为各种决策支持和科学研究提供快捷有效的数据挖掘解决方案，以数据仓库作为数据平台为数据挖掘提供数据来源，以数据挖掘工具作为系统重心，面向任务组织数据挖掘流程，这种方案是可行的并且具有良好的应用价值。

同时，我们在设计时充分考虑了开放性和可伸缩性，使系统能够适用于各种不同的情况。并且，我们设计了一整套面向对象的元数据，以这些元数据为核心控制管理整个系统，充分保证了系统的一致性、灵活性和可维护性。

13.5 元数据管理

元数据就是关于数据的数据。在数据仓库中，元数据具有重要的地位，对数据仓库的设计、开发、维护和管理，对数据的组织、信息的查询和结果的理解都有重大作用。根据元数据在数据仓库周期中的作用，可以将其分为技术级元数据和商业级元数据。技术级元数据主要服务于信息技术环境的开发、维护和管理过程中的设计者、开发者和管理者。企业级元数据服务于企业的运作环境，使它更有益于终端用户的理解。

在 MSMiner 中，我们拓展了元数据的作用范围，用元数据描述和管理整个系统的数据和环境，不仅包括数据仓库平台中的数据，并且包括数据挖掘工具中的任务模型。这样，元数据居于整个系统的核心地位，统一管理数据仓库和数据挖掘工具，并控制整个数据挖掘流程，包括数据准备、采掘、表述以及评价，使数据和数据挖掘任务有机地结合在一起。

同时，MSMiner 采用了一种面向对象的方法对复杂的元数据进行管理和维护，保证了元数据的一致性和系统的健壮性。

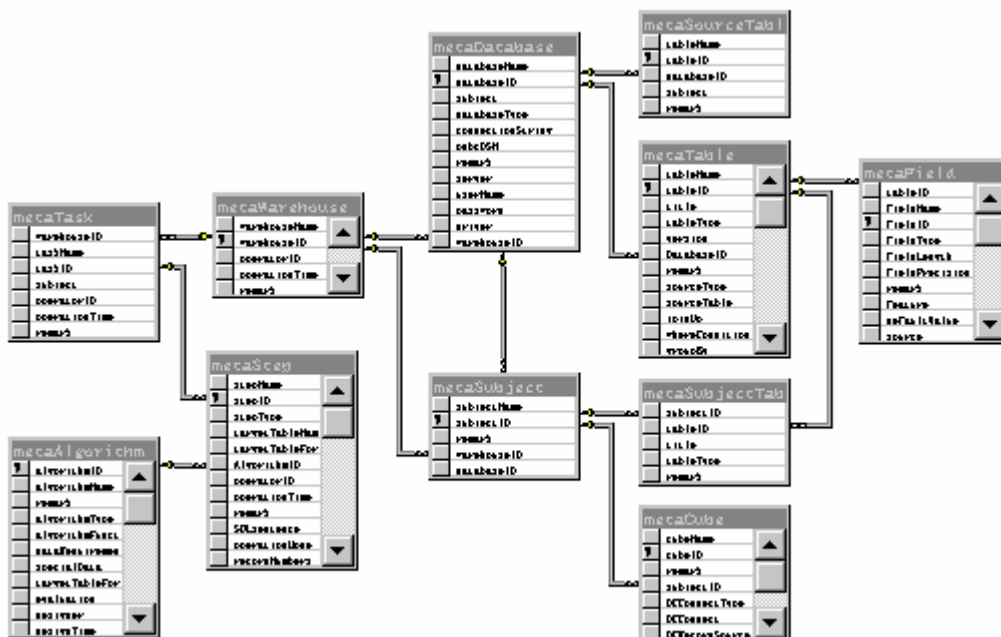
13.5.1 MSMiner 元数据的内容

MSMiner 系统的元数据主要包括以下几方面内容：

1. 外部数据源描述。包括数据源连接及环境信息，数据源内容的注册和描述，包括外部数据库及其表、字段等的描述。

- ### 13.5.2 MSMiner 元数据库

为了保证元数据的一致性，这些元数据表互相关联，遵循一定的约束关系。例如，`metaDatabase` 和 `metaTable` 这两个元数据表通过 `databaseID` 互相关联，当修改或删除 `metaDatabase` 表中的一条记录时，`metaTable` 中的相关记录必须做相应改动。图 13.4 显示了几个主要的元数据表之间的约束关系：



335

附加约束关系的元数据表只是元数据内容的存储方式，用户并未直接与这些表打交道，而是通过一种面向对象的元数据模型来操纵这些数据。

13.5.3 MSMiner 元数据对象模型

MSMiner 的元数据数量庞大并且关系复杂，很自然的想法是，通过一种相对独立的模块专门负责对元数据的存取管理，这个模块应该符合以下设计要求：

- 一致性。
对元数据的操作应保证元数据的一致性和完整性。对某一元数据进行修改时，必须使其他相关元数据同步更新。
- 完备性。
能够对所有元数据进行统一管理，满足任何元数据操作的需要。
- 易维护性。
通过该模块对元数据的操作管理应逻辑清晰、简便易行。

根据以上设计要求，我们设计了一种面向对象的元数据结构。即将各种元数据封装在相应的元数据类中，将这些元数据类的对象实例通过层次结构有机地组织起来，构成一种层次型对象模型。系统通过这些对象对各元数据表进行操作管理。

系统通过这种对象模型访问元数据，而不需要直接接触元数据库。在经过良好封装的元数据类中包含各种属性和方法，属性表达了相应的元数据值，而方法定义了对相关元数据的各种操作，并负责维护元数据之间的一致性。元数据的存取、更新和管理通过访问这些属性和方法来实现。

MSMiner 的元数据对象模型的设计不仅保证了元数据的一致性，并且使系统建立和维护工作的难度大大降低，增强了系统的健壮性和可操作性。

MSMiner 的元数据类在实现上分为两种：个体类和集合类。

个体类主要包含对应相关元数据表中各个元数据值的各种属性，负责存取和更新相关元数据表中的单条记录。

集合类包含了相关个体类对象的集合，负责对整个元数据表进行存取和更新以及维护各类元数据之间的一致性。

集合类和个体类的各个对象实例按照一定的逻辑关系互相包含，构成了如图 13.5 所示的树形结构对象模型。

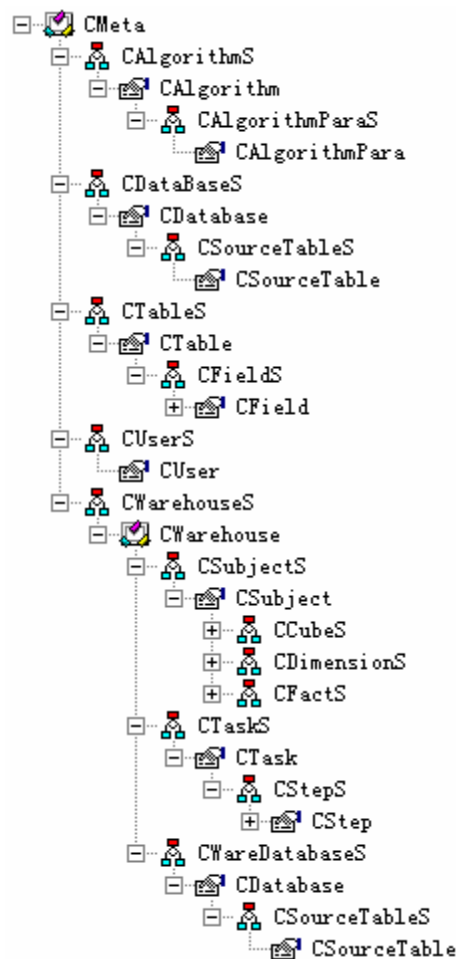


图 13.5 元数据对象模型

对元数据进行更新时,直接更新相应个体类对象中的相关属性,然后调用该对象的 Update()方法完成对元数据表的操作,同时触发对其他相关元数据对象的更新。

对元数据进行增删操作时,通过调用相应集合类对象中的 Add()或 Remove()方法,增加或删除集合中的个体对象并更新元数据表,同时触发对其他相关元数据对象的更新。

通过这种方式,不仅逻辑清晰地表达了整个元数据的结构,并且有效地保证了元数据的一致性。

各个集合类中的属性和方法基本类似,主要有以下几种:

Item(index)	获得个体对象集合中的一个对象
Count	个体对象集合中的对象总数
Add(object)	新增一个个体对象
Remove(index)	删除一个个体对象
GetCollection()	初始化时,从元数据表中读取得到个体对象集合

各个个体类中的属性和方法与相关元数据的内容有关,以下将具体说明一些主要的元数据个体类的设计。

MSMiner 元数据的所有内容的描述,都包含在一个主元数据类 CMeta 之中。CMeta 类的定义如下:

```

CMeta
{

```

Algorithms	算法集合类 CAlgorithmS 的对象实例
Databases	数据库集合类 CDatabaseS 的对象实例
Tables	表集合类 CTableS 的对象实例
Users	用户集合类 CUserS 的对象实例
Warehouses	数据仓库集合类 CWarehouseS 的对象实例

}

其中, Algorithms 包含了算法个体类 CAlgorithm 的对象实例集合, 注册了各种数据挖掘算法的详细信息。

Databases 包含了数据库个体类 CDatabase 的对象实例集合, 描述各个外部数据库的详细信息。

Tables 包含了数据表个体类 CTable 的对象实例集合, 描述所有注册外部数据表和数据仓库内部数据表的详细信息。

Users 包含了用户个体类 CUser 的对象实例集合, 登记各用户的详细信息。

Warehouses 包含了数据仓库个体类 CWarehouse 的对象实例集合, 描述数据仓库中各种内容的详细信息。

这五个元数据集合对象基本包含了所有元数据的内容, 其他各种元数据对象都按照一定的关系包含在五个对象之中。系统运行时生成 CMeta 的一个对象实例, 在这个对象中就包含了其他所有元数据对象的实现。通过这个对象逐层深入, 就可对所有元数据进行分类存取和管理。

以下是数据仓库类 CWarehouse 的定义:

CWarehouse

```
{
    WarehouseID      数据仓库 ID
    WarehouseName     数据仓库名称
    Subjects          主题集合类 CSubjectS 的对象实例
    Tasks             数据挖掘任务集合类 CTableS 的对象实例
    WareDatabases     数据库集合类 CDatabaseS 的对象实例
    ... ..
}
```

其中, Subjects 包含了主题个体类 CSubject 的对象实例集合, 描述各个主题及其所包含的事实表、维表、中间表和数据立方体的详细信息。

Tasks 包含了数据挖掘任务个体类 CTask 的对象实例集合, 描述各个数据挖掘任务及其各步骤的详细信息。

WareDatabases 包含了数据库个体类 CDatabase 的对象实例集合, 描述数据仓库中各个内部数据库的详细信息。

元数据类 CWarehouse 的结构反映了 MSMiner 系统的设计思路: 数据仓库作为数据挖掘的数据平台, 按主题组织数据, 为数据挖掘任务提供数据来源。整个系统的结构即按照 CWarehouse 类的结构来组织。

MSMiner 元数据的管理是在各功能模块的管理维护中实现的。在通过 MSMiner 各个功能模块对数据仓库各部分和数据挖掘任务进行建立、修改和日常维护时, 元数据中的相应内容也同时得到新增、修改和维护。用户在管理数据仓库和数据挖掘任务的同时, 也就是在管理元数据。实际上, 用户就是通过在各个功能模块中对后台元数据进行维护来实现对整个系统的管理。

对元数据各部分内容的管理, 分布在系统中的数据抽取和集成模块、主题组织模块、OLAP 模块、数据挖掘任务编辑模块及算法管理模块等各个模块之中。

元数据在数据仓库中占有极其重要的地位，而 MSMiner 进一步拓展了元数据的范围，使它对从数据抽取、数据管理到数据挖掘的整个过程进行统一的管理和控制，元数据成为整个系统的核心和灵魂。

为了保证对复杂元数据的有效管理和元数据的一致性，我们设计了一种元数据对象模型，将元数据的存取和管理封装在多个元数据类中，通过互相关联的多个元数据对象对元数据进行操作，保证了整个系统的一致性和可维护性。

13.6 数据仓库

13.6.1 数据仓库含义

到八十年代，计算机技术的迅猛发展已使大部分企业在他们的数据库中积累了大量的数据，人们已不再满足于简单的数据操作，产生了进一步使用现有数据的需求，也就是利用现有的数据，进行分析和推理，从而为决策提供依据。这种需求既要求联机服务，又涉及大量用于决策的数据，而传统的数据库系统已无法满足这种需求。其具体体现在三个方面：①决策所需历史数据量很大，而传统数据库一般只存储短期数据。②辅助决策信息涉及许多部门的数据，而不同系统的数据难以集成。③由于访问数据的能力不足，它对大量数据的访问性能明显下降。

随着 C/S（客户/服务器）技术的成熟和并行数据库的发展，信息处理技术的发展趋势是：从大量的事务型数据库中抽取数据，并将其清理、转换为新的存储格式，即为决策目标把数据聚合在一种特殊的格式中。随着此过程的发展和完善，这种九十年代初出现的支持决策的、特殊的数据存储即被称为数据仓库（Data Warehouse）。

目前，人们对数据仓库有很多不同的理解，但都有共同之处，Inmon 将数据仓库明确定义如下[Inmon 1992]：

数据仓库（Data Warehouse）是面向主题的、集成的、内容相对稳定的、不同时间的数据集合，用以支持经营管理中的决策制定过程。

数据仓库收集存于不同数据源中的数据，将这些分散的数据集中到一个更大的库中，最终用户从数据仓库中进行查询和数据分析。数据仓库中的数据应是良好定义的、一致的、不变的，数据量也应足够支持数据分析、查询、报表生成和与长期积累的历史数据的对比。数据仓库是一个决策支持环境，通过数据的组织给决策支持者提供分布的、跨平台的数据，使用过程中可忽略许多技术细节。

在数据仓库的基础之上，可建立多种分析决策工具，如 OLAP（在线分析处理）和数据挖掘工具等，以提供各种决策支持服务。

数据仓库有四个基本特征：数据仓库的数据是面向主题的；数据仓库的数据是集成的；数据仓库的数据是不可更新的；数据仓库的数据是随时间不断变化的。

(1) 数据仓库中的数据是面向主题的。

这是与传统数据库面向应用相对应的。主题是一个在较高层次将数据归类的标准，每一个主题基本对应一个宏观的分析领域。比如，一个保险公司的数据仓库所组织的主题可能为：客户，政策，保险金，索赔。而按应用来组织则可能是：汽车保险，生命保险，健康保险，伤亡保险。基于主题组织的数据被划分为各自独立的领域，每个领域有自己的逻辑内涵互不交叉。主题域是独立和完备的，适用于分析型的应用。而基于应用的数据组织则完全不同，它的数据只是为处理具体应用而组织在一起的。应用是客观世界既定的，它对于数据内容的

划分未必适用于分析所需。

(2) 数据仓库的数据是集成的。

在数据进入数据仓库之前，必然要经过加工与集成。这一步实际上是数据仓库建设中最关键、最复杂的一步。首先，要统一原始数据中的所有矛盾之处，如字段的同名异义、异名同义、单位不统一、字长不一致等等，还要进行数据综合和计算，总之要将原始数据结构做一个从面向应用到面向主题的转变。

(3) 数据仓库的数据是稳定的。

数据仓库的数据主要供决策分析之用，所涉及的操作主要是数据查询，一般不进行修改操作。数据仓库的数据反映的是一段相当长的时间内历史数据的内容，是不同时间点的数据快照的集合，以及基于这些快照进行统计、综合和重组的导出数据，而不是联机处理的数据。

(4) 数据仓库的数据又是随时间不断变化的。

数据仓库的数据不是实时更新的，但并不是永远不变的，也要随着时间的变化不断地更新、增删和重新综合。它表现在以下几个方面：首先，数据仓库内的数据时限要远远长于操作型环境中的数据时限。数据仓库保存数据时限较长是为了适应 DSS 进行趋势分析的要求。其次，操作型环境包含当前数据，即在存取一刹那正确、有效的数据；而数据仓库中的数据都是历史数据。最后，数据仓库数据的码键都包含时间项，从而标明了该数据的历史时期。

从信息技术的观点来看，数据仓库关心的是在一个组织机构中将适当的信息传送到适当的个人手中。这是一个正在进行的过程，而不是以前的解决方案，并且需要不同的方法以满足面向事务系统开发工作的需求。

数据仓库是数据的集合，以对那些面向主题的、集成的、时间不同的、非易失性的决策的管理工作给以支持。数据仓库关注的是概念（如销售），而不是过程（如提供发票）。它包括从多种处理系统收集到的有关某一概念的所有相关信息。信息进行定期收集和存储，并且是相对稳定的。

通过一致命名约定、测量、物理属性和语义，数据仓库对操作数据进行集成。在数据仓库物理设计的第一步是确定应包括哪些主题领域，并开发一套意见一致的定义集。这需要约见最终用户、分析员、高级管理人员，以了解所需信息的范围，并给出相应文档。在将逻辑处理转化成物理数据仓库之前，必须对相关问题有彻底的了解。

MSMiner 中通过数据仓库管理器为数据挖掘工具提供数据平台，其主要目标是通过建立和维护数据仓库，为数据挖掘提供经过清洗、整合、转换的数据来源，完成数据挖掘过程中数据预处理的部分任务。

MSMiner 数据仓库中的数据按主题组织，主题中包含多个事实表、维表、中间表以及数据立方体。数据仓库主题按星型模式建模，在此基础上实现多维数据立方体和各种 OLAP 操作，并为数据挖掘任务提供经过适当预处理和良好组织的数据源，最后结果可由可视化工具显示，或以报表的形式输出。整个数据仓库，包括数据的抽取及主题的组织等，均通过元数据来管理和维护。

13.6.2 MSMiner 数据仓库的基本结构

MSMiner 数据仓库主要包括以下四个功能模块：数据抽取和集成模块、主题组织模块、OLAP 及可视化模块。

数据抽取和集成模块是 MSMiner 数据仓库中的一个非常重要的组成部分，负责从外部数据源中抽取数据，并集成到主题的各种综合数据表中。模块通过 OLE DB for ODBC 接口与各

种关系型数据库连接。抽取数据时选取合适的数据库，指定它所在的数据库和表，经过一定的清洗、聚集、转换处理后，将数据集成到数据仓库中的各种目标表中。

主题组织模块负责星型模型的组织和对事实表、维表和中间表的管理。

OLAP 模块负责在星型模式的基础上建立数据立方体，实现各种多维数据分析和查询操作。

可视化模块提供多种可视化手段，通过各种三维及二维图表等直观地显示数据；并专门针对各种数据挖掘算法，提供相应的可视化手段表示数据挖掘的结果。

MSMiner 数据仓库的结构示意图如图 13.6 所示。

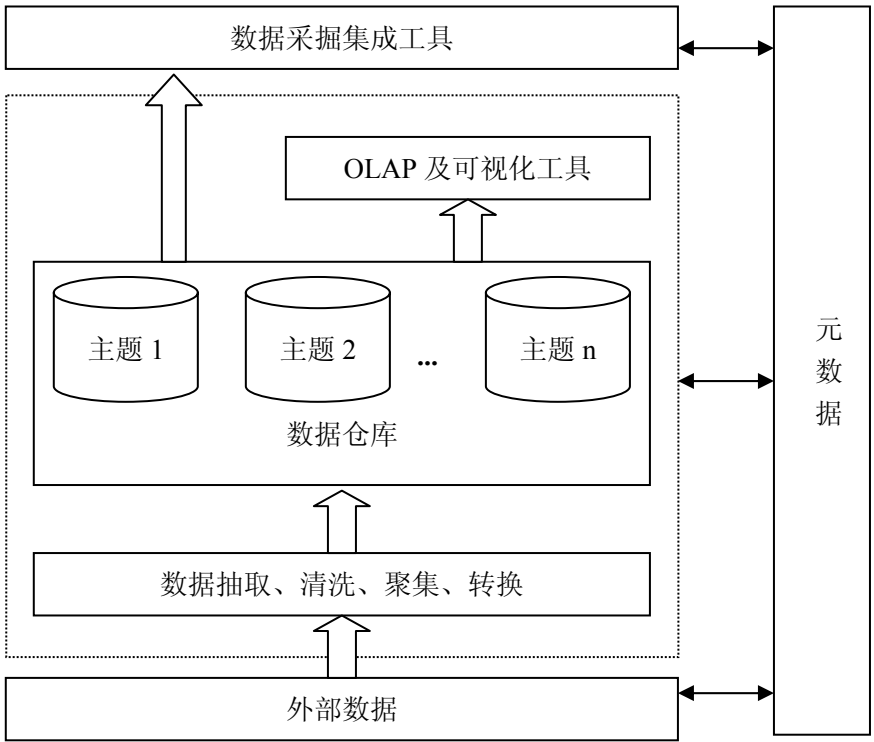


图 13.6 MSMiner 数据仓库结构示意图

目前多数数据仓库都在传统的关系型数据库管理系统 RDBMS 上实现。建立在关系型数据库上可利用各种已有的成熟的数据库技术，使数据仓库的构建更加便捷有效。但传统的 RDBMS 实现往往是为事务型数据处理做优化的，数据仓库的很多特性，例如处理大规模历史性和综合性数据，灵活的多维数据查询等，会因此受到限制。现在已经出现了多种不同的技术途径，以改进数据仓库的数据库解决方案，例如：

- 基于并行计算平台的并行关系型数据库。
- 使用新的索引技术提高传统 RDBMS 的性能。
- 基于专门的数据库技术或已熟知的 RDBMS 的多维数据库 MDDB。这种解决方案往往与 OLAP 技术紧密集成。

根据 MSMiner 的设计目标，我们更注重实现数据仓库的易用性和集成性，所以 MSMiner 数据仓库的构建仍是基于传统的关系型数据库管理系统，其核心数据库由 Microsoft SQL Server 7.0 DBMS 实现。

Microsoft SQL Server 7.0 提供了许多支持数据仓库的特性，例如功能强大的 OLAP 服务器等。即将发布的 SQL Server 2000 更是增加了许多先进的功能，包括一套完全的数据库和数据分析解决方案，一个新的集成的数据挖掘引擎等。MSMiner 的进一步工作将充分利用这些有

用的特性，增强系统功能。

MSMiner 数据仓库的具体实现包含多个数据库，这些数据库分为两类：其中一个数据库为元数据库，专门存储各种元数据表；其他数据库为主题数据库，分别存储各个主题中的各种综合数据表。

MSMiner 的外部数据源也主要是各种关系型数据库。MSMiner 通过 OLE DB for ODBC 接口，可与各种流行的数据库产品连接，包括 Oracle、Sybase、Microsoft 等众多厂商的大型数据库系统和小型的桌面数据库产品，保证了系统的开放性。OLE DB 定义了一个 COM 接口的集合，它封装了各种数据库管理系统服务。这些接口允许创建实现这些服务的软件组件。OLE DB 组件包括数据提供者（它们存储和发布数据）、数据用户（它们使用数据）和服务组件（它们处理和传输数据）。OLE DB 接口用来平滑地集成组件，以便供货商能迅速地将高质量的 OLE DB 组件推向市场。另外，OLE DB 包括连接 ODBC 的桥梁，这使得对现在大量的 ODBC 关系数据库提供持续支持成为可能。

MSMiner 也可从多种数据文件中获取数据，并将实现 Web 数据的获取。本文以下所指的外部数据源将主要指各种关系型数据库。

13.6.3 主题

MSMiner 数据仓库以主题（Subject）为基本组成单位。一个主题存储在一个 SQL Server 数据库中。每个主题包含多个事实表、维表、中间表以及数据立方体，它们在本质上都是一些综合数据表。用户根据需要，通过数据抽取模块，从外部数据源中抽取数据，并经过各种清洗、聚集和转换处理后，将数据集成到数据仓库主题的各个表中。

事实表（Fact）存储用户需要查询分析的数据，主题的主要内容就在事实表中体现。事实表中一般包含多个维（Dimension）和度量（Measurement）。

维代表了用户观察数据的特定视角，如：时间维、地区维、产品维等。每一个维可划分为不同的层次来取值，如时间维的值可按年份、季度、月份来划分，描述了不同的查询层次。

度量是数据的实际意义，描述数据“是什么”，即一个数值的测量指标，如：人数、单价、销售量等。

度量的实际数据存放在事实表中。维的详细信息，如不同的层次划分和相应数据等在维表中存储，事实表中存放各个维的标识码键。事实表和维表将通过这些键关联起来，构成一种星型模型，在此基础上构成数据立方体并实现 OLAP 操作。

中间表是一种面向数据挖掘任务的综合数据表。特定的数据挖掘任务往往需要特定的、经过一定预处理的整合数据，这些数据存放在中间表中，供数据挖掘任务使用。数据挖掘的结果也可存放在中间表中。

数据立方体（Data Cube）是一个多维的数据阵列结构，是组织多维 OLAP 操作的基础。在 MSMiner 的实现中，数据立方体实际上是一种逻辑概念结构，它的实现是基于事实表和维表关联的星型模型，具体体现为主题中的一张表。通过在主题中选取所需的数据维和度量，生成描述星型模型关联信息的元数据表，数据立方体就是该表的逻辑抽象。

13.6.4 数据抽取和集成

数据仓库中的数据是集成的，是从原有的分散的数据库中抽取出来的。在各个源数据库中的操作型数据和数据仓库中的分析型数据有很大差别，数据仓库中的数据不能从原有的数

数据库系统中直接得到，必须要经过清洗、聚集和转换。这是建设数据仓库过程中非常关键而复杂的一步。所要完成的工作有：

- 在各个分散的数据库系统中的源数据往往有很多重复或不一致的地方，要统一源数据中的所有矛盾之处，如字段的同名异义、异名同义、单位不统一、字长不一致等。
- 得到数据仓库中的综合数据，需要将原始数据结构做一个从面向应用到面向主题的转化，进行必要的综合和计算。

MSMiner 提供专门的数据抽取和集成模块来完成上述工作。数据抽取和集成工作在建立主题后进行，主要为事实表、维表和数据挖掘中间表的生成提供整合数据。

模块主要提供三个层次的功能：数据的简单抽取和集成，数据的复杂处理和面向数据挖掘的数据预处理。

13.6.4.1 数据的简单抽取和集成

在进行数据抽取工作之前，必须先注册外部数据源。这里的外部数据源主要是各种关系型数据库，系统通过 OLE DB for ODBC 接口与这些数据源连接，并将数据源的详细信息注册在元数据中。这些信息包括外部数据库的 ODBC 连接参数，数据表信息以及各个表中的字段信息等。

抽取数据时，以数据表为基本存储单位，抽取步骤如下：

1. 指定数据源表及目标表。数据源表可以是不同的多个表，目标表根据用户命名在指定主题中新建，表的结构根据以下字段定义步骤决定。
2. 定义目标表中的字段及数据来源。具体包括以下几个步骤：
 - (a) 从数据源表中选取一个或多个源字段。
 - (b) 为源字段指定简单数据集成操作，包括各种数学运算、日期运算、字符串操作等。
 - (c) 指定目标字段名。目标字段的类型、精度、长度等由源字段及集成操作决定。
3. 重复步骤 2，直至所需的字段选取和定义完毕。
4. 指定数据源表之间的关联，设置 OrderBy 和 GroupBy 等参数，加入其他抽取条件等。
5. 根据以上定义，动态生成数据抽取 SQL 语句，从源数据库中将数据抽取到数据仓库指定主题的目标表中。

例如，位于外部服务器 Billy 上的 SQL Server 数据库 TaxData 为已注册外部数据库，我们从其中的数据表 Tax 和 TaxDate 中抽取数据，在数据仓库中生成事实表 Fact1，如图 13.7 所示。

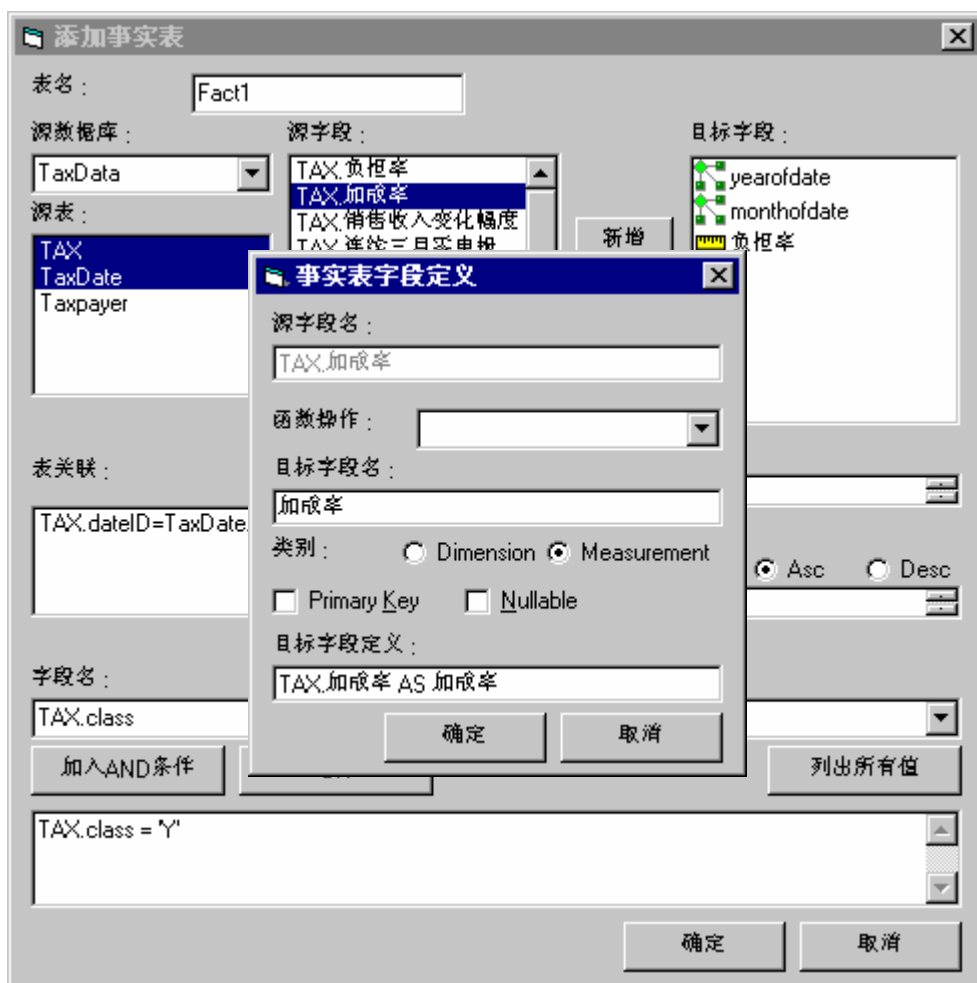


图 13.7 数据抽取和集成

定义四个目标字段分别为：

度量： 加成率 [TAX.加成率 AS 加成率]

度量： 负担率 [TAX.负担率 AS 负担率]

维： yearofdate [Year(TAXDate.date) AS yearofdate]

维： monthofdate [Month(TAXDate.date) AS monthofdate]

设置数据源表之间的关联为：

TAX.taxID=TaxDate.dateID

加入抽取条件：

TAX.class = 'Y'

最后系统将生成如下 SQL 语句，生成新的事实表并完成数据抽取：

```
SELECT DISTINCT TAX.加成率 AS 加成率,
                TAX.负担率 AS 负担率,
                Year(TAXDate.date) AS yearofdate,
                Month(TAXDate.date) AS monthofdate
INTO Fact1
FROM Billy.TaxData.dbo.TAX AS TAX JOIN
     Billy.TaxData.dbo.TAXDate AS TAXDate ON
     TAX.dateID=TaxDate.dateID
```

WHERE TAX.class = 'Y'

目前的系统实现要求对于每次数据抽取，各源数据表必须在同一个数据库中。如果要求跨数据源抽取数据，可以先为不同的数据源生成若干个临时表或视图，再将这些临时表或视图整合为一张目标表。

12.5.3.2 数据的复杂处理

数据的简单抽取和集成功能已基本能满足系统大部分需要，但对于各种不一致数据、错误数据以及复杂的数据集成，往往无法用一句 SQL 语句完成，需要更进一步的工作。所以系统提供复杂数据处理功能，以完成各种数据清洗、转换和复杂集成工作。

在数据抽取和集成模块中，用户定义并进行简单数据抽取后，可对生成的数据表进行进一步处理。系统提供专门接口，由用户针对特殊的转换处理编写专门的 SQL 语句，这种处理语句可以有一句或多句，由系统自动控制顺序执行，经一系列视图或临时表的转换之后，完成复杂数据处理任务。

上述所有数据抽取和集成处理操作都将存入元数据统一管理。当需要重新抽取数据，或进行数据更新时，系统将根据元数据中记录的数据抽取和集成操作步骤，依次完成数据的抽取和集成工作。

需要指出的是，上述各种数据抽取和处理步骤的顺序并不是一定的，各个步骤所要完成的具体工作以及各个步骤的处理顺序依赖于所抽取和处理的数据的特性。

经过上述的处理工作，基本上已经能够满足绝大部分数据处理需求，但我们面临的数据将是非常复杂和多样的，数据抽取和集成工作也会出现多种多样的复杂情况，对于有些情况通过上述的步骤可能仍然难以处理。这时，我们将提供对数据表的直接编辑界面，由用户手动地直接对表的格式和数据进行更进一步的处理工作。这种对数据表的直接修改将不在元数据中体现。

12.5.3.3 面向数据挖掘的数据预处理

MSMiner 数据仓库的一个主要的设计目标是为数据挖掘提供适当数据。很多数据挖掘任务需要符合某种格式，并且经过适当预处理的数据。为此数据抽取和集成模块提供了专门的数据预处理功能。

如果经过前面的数据抽取和集成得到的数据仍不能满足数据挖掘的需要，需进一步做某些特殊的预处理，那么可以在数据挖掘算法库中选择适当的算法，对数据进行深层次的处理，得到所需的数据。如利用粗糙集算法对属性集进行约简等。数据预处理所用的算法同样作为一种数据抽取和集成规则存入元数据中。

这里的数据预处理功能相当于数据挖掘任务中一个简化的步骤。实际上，用户也可以不在数据抽取和集成的过程中，而在数据挖掘任务的步骤之中进行数据预处理工作。具体视数据预处理的复杂程度和情况的需要而定。

13.6.5 数据抽取和集成的元数据

数据的清洗、转换和整合工作，需通过建立完备的元数据来进行管理。建立数据仓库，进行数据抽取和集成时，生成各种数据抽取和处理规则，并存储在元数据中，数据仓库的数

据抽取和更新都将在这些元数据的指导下进行，并根据决策需求的变化及时维护和更新这些元数据。

这些数据抽取和处理规则分布在描述数据仓库各个表及其字段信息的元数据中，主要由元数据类 CTable 和 CField 来管理。这两个元数据类的定义如下：

```
CTable
{
  Attributes:
    TableID          表的标识 ID
    TableName        表名
    TableType        表的类别（事实表、维表、中间表或
                    外部数据源表）
    DatabaseID       所属数据库 ID
    Fields           该表所包含的字段集合
    SQLSentence      数据抽取 SQL 语句
    AlgorithmID      用于数据预处理的算法 ID
    Remark           表的说明
    ... ..
  Methods:
    Update()
    ... ..
}
```

元数据类 CTable 中的 Fields 是元数据集合类 CFieldS 的对象实例，其中包含元数据个体类 CField 的对象实例的集合，记录所描述数据表包含的字段的详细信息。

From、Join、Where、OrderBy、GroupBy 等分别记录了定义数据抽取规则时，用以生成 SQL 抽取语句的各有关部分的定义。SQLSentence 记录了完整的 SQL 抽取语句，由以上各部分定义结合各字段的数据来源定义共同拼合生成。对于复杂数据抽取，SQLSentence 可能包含多个 SQL 语句，此时上面各个子部分的定义无效。

AlgorithmID 记录了进行面向数据挖掘的数据预处理时所用算法的 ID，算法存储在数据挖掘算法库中，并在元数据中详细注册。

```
CField
{
  Attributes:
    FieldID          字段 ID
    FieldName        字段名
    TableID          所属表 ID
    FieldType        字段数据类型
    Source           字段的数据来源定义
    Remark           字段的说明
    ... ..
  Methods:
    GetTableInfo()  获得所属表的详细信息
    Update()
    ... ..
}
```


元数据类 CField 中的 Source 属性记录了该字段的详细定义，包含了数据来源字段及其简单集成操作定义。

通过 CTable 和 CField 元数据类，可以详细注册数据仓库中各个表及其字段的定义，以及详细的数据抽取信息。数据的抽取和更新主要通过这两个元数据类来维护。

13.6.6 数据仓库建模及 OLAP 的实现

13.6.6.1 星型模型

MSMiner 数据仓库按照星型模型（Star Schema）建模。每个主题中包含多个事实表和维表。以一个事实表为中心，关联多个维表，构成一个星型结构。多个星型结构共同组成了一个主题。图 13.8 显示了星型模型。

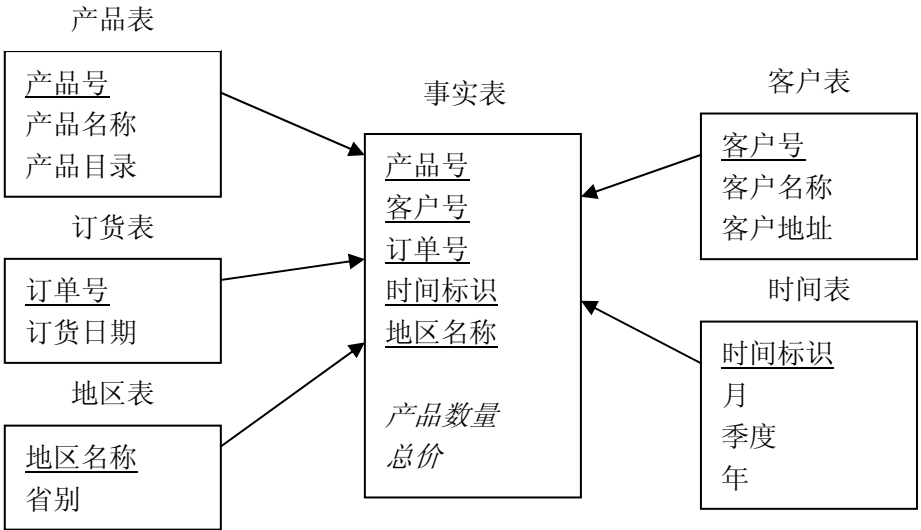


图 13.8 星型模型

MSMiner 数据仓库中创建主题并建立星型模型的步骤如下：

1. 用户指定主题名称和其他说明信息，系统生成一个空主题及相应的数据库，此时主题中没有任何内容，即主题对应数据库中不含任何数据表。用户按以下步骤逐步生成维表和事实表，并建立星型模型。
2. 选择适当的数据源，调用数据抽取和集成模块，生成多个维表。维表中包含维的层次和取值等信息。每个维表中必须包含一个标识码键作为主键。
3. 在生成多个维表之后，再调用数据抽取和集成模块，从适当的数据源中抽取数据生成事实表。每个事实表中包含多个维字段和度量字段。度量字段包含将要实际分析使用的数据。每个维字段与一个维表中的标识码键相关联，从而使多个维表与事实表关联构成一个星型模型。

13.6.6.2 数据立方体

数据立方体（Data Cube）是将数据按多个维度组织形成的一种多维结构。用户可以灵活

地从各个角度进行切片、旋转等操作，获得所需的数据。如图 12.8 所示。

MSMiner 的 OLAP 在关系型数据表上实现，即采用 ROLAP 方式。在主题中事实表关联多个维表的星型结构之上模拟多维数据立方体，然后在数据立方体之上实现 OLAP 操作。各种 OLAP 操作请求将通过一个 ROLAP 引擎动态翻译成 SQL 语句，在数据仓库中的综合数据表（事实表）中进行查询得到结果。

由于数据立方体在表现上缺乏直观性，尤其当维度超出三维后，数据的采集和表示都比较困难，因此我们采用一种二维表的结构来表示整个数据立方体，并在此之上进行 OLAP 操作。数据立方体中的维都被划分到行维或列维两种维度中去，在行维和列维交织成的二维表中显示相应的度量数据，实现以二维表格反映多维特征，提供数据立方体的可视化表示。

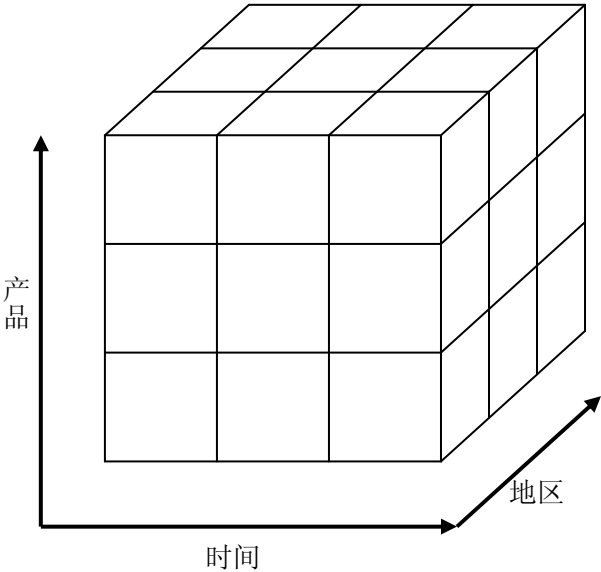


图 13.9 数据立方体

数据立方体的建立步骤如下：

1. 从指定主题中选择一个事实表。
2. 从事实表中选择若干个维和度量。

维必须被指定为行维或者列维。如果有的维包含多个层次，它的各个层次也可作为多个子维选入，以实现向上综合和向下钻取功能。

用户的兴趣往往不仅在于获得已有的数据，还希望能够得到综合数据。因此对度量需指定某种运算操作，如求和、均值、极值等。

3. 生成数据立方体。

表 13.1 是一个赢利额关于时间维、产品类型维和销售地区维的简单 OLAP 示例。时间维包括两个细节项：(1995, 1996)，产品类型维包括三个细节项：(Software, Hardware)，而销售地点维包括四个细节项：(East, West, North, South)。整合后的关系型数据表有如下信息：

表 13.1 赢利额

Time	Division	Region	Revenue
1995	Hardware	East	120
1995	Hardware	West	150
1995	Hardware	North	195
1995	Hardware	South	165
1996	Hardware	East	180
1996	Hardware	West	152

1996	Hardware	North	215
1996	Hardware	South	190
1995	Software	East	200
1995	Software	West	185
1995	Software	North	130
1995	Software	South	134
1996	Software	East	250
1996	Software	West	204
1996	Software	North	154
1996	Software	South	146

从上表中选取时间维、产品类型维和销售地区维作为三个维，选取赢利额作为度量并对赢利额求和，建立数据立方体，并以二维表形式表示如表 13.2。

表 13.2 二维表

		East	West	North	South	Total
1995	Hardware	120	150	195	165	630
	Software	200	185	130	134	649
1996	Hardware	180	152	215	190	737
	Software	250	204	154	146	754
Total		750	691	694	635	2770

在这种以二维表形式表示的数据立方体上，可以方便地进行各种多维数据分析查询操作。如用户可以展开产品维，查看各个销售地区 1995 年各类产品的赢利额（向下钻取）；也可以合并产品维，查看各个销售地区 1995 年所有产品的总赢利额（向上综合）；或改变查询角度，查看各类产品 1995 年在所有地区的总赢利额（旋转）等等。

MSMiner 提供了一个良好的用户界面以二维表形式显示数据立方体，用户只需通过点击（Click）、拖放（Drag and Drop）等简单动作就可以进行各种可视化的多维分析操作，并通过多种三维或二维图表的可视化方式显示结果。如图 13.10 所示。

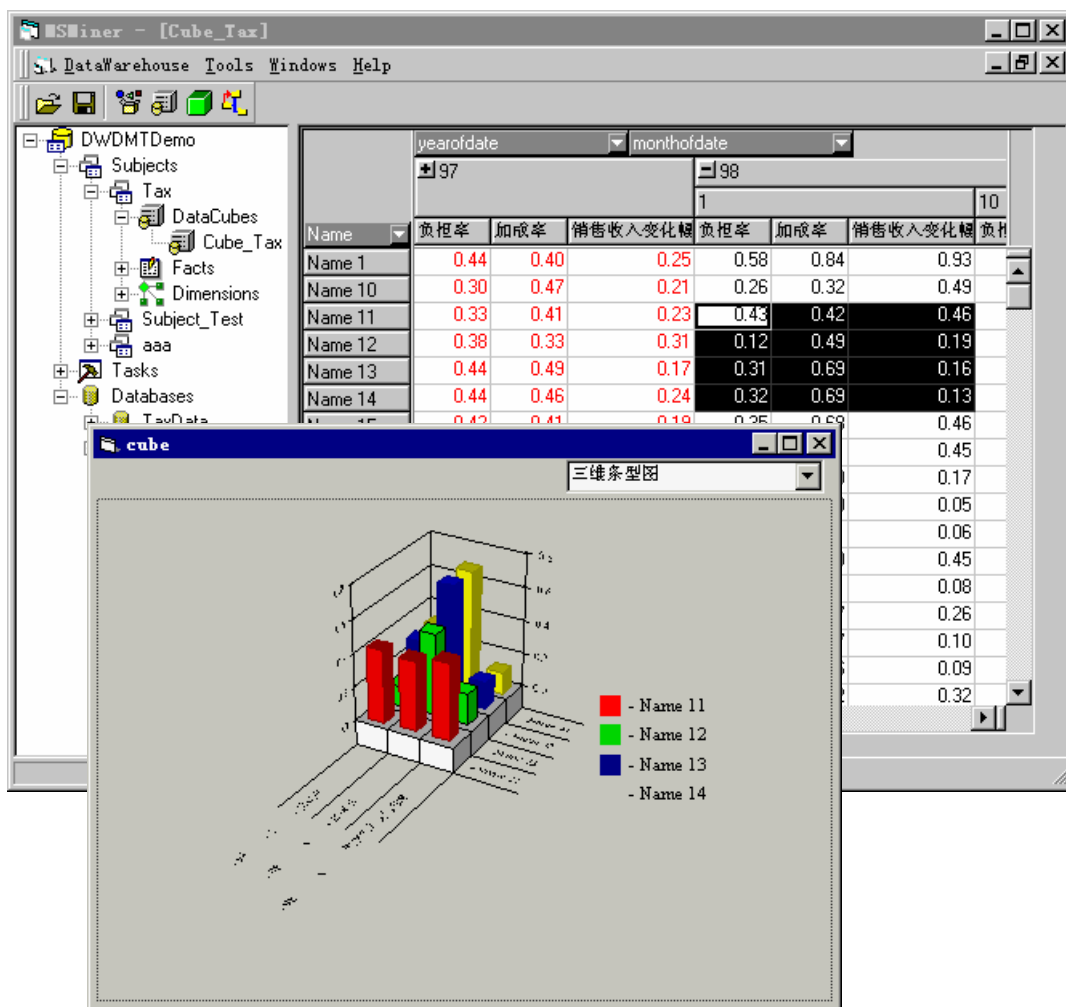


图 13.10 MSMiner 数据立方体及 OLAP

13.6.6.3 多维数据分析

多维分析是指对以多维形式组织起来的数据采取切片、切块、旋转、钻取等各种分析动作，以求分析数据，使用户能从多个角度、多侧面地观察数据库中的数据，从而深入了解包含在数据中的信息、内涵。多维分析的基本操作有：

- 切片（Slice）。在多维数组的某一维上选定一维成员或选定多维数组的一个二维子集的动作叫作切片。
- 切块（Dice）。选定多维数组的一个三维子集的动作称为切块。
- 旋转（Pivot）。旋转就是改变数据立方体的维方向。例如，交换两个行维的位置，或是把某一个行维移到列维中去，或是把页面显示中的一个维和页面外的维进行交换（令其成为新的行或列中的一个）。
- 向上综合（Roll Up）。将数据综合到较高的维层次。
- 相下钻取（Drill Down）。将数据细化至较低的维层次。

用户在数据立方体的二维表界面上进行多维数据分析操作时，各种操作信息通过 OLAP

引擎动态转化为 SQL 数据查询请求，在数据仓库中查询获得结果数据。具体实现步骤如下：

根据用户操作，确定所涉及的维（分组指标）和度量（统计指标）。

ROLAP 引擎将用户查询和报表请求翻译成一个或多个 SQL 语句。

系统首先在多维立方体的元数据中查询有关统计指标和分组指标的内容，生成相关 SQL 语句。在元数据中查询有关统计指标和分组指标的内容时，先找统计指标，分组指标如没有找到，找到比它层次低的分组指标亦可。但这时需再作相关汇总和计算。

对于无法用一个 SQL 语句生成的查询，采用多级数据缓冲（临时表或视图）的方法解决。

ROLAP 引擎将 SQL 语句提交数据库或数据仓库，提取数据后进行相应处理和计算，然后再将结果集返回给前台显示。

小结：

MSMiner 数据仓库以主题为基本构成单位，主题中包含事实表、维表和各种中间表。MSMiner 数据仓库作为数据挖掘任务的数据平台，提供有效手段从各种外部数据源中抽取和集成数据，完成数据挖掘过程中前期数据准备阶段的部分工作。

MSMiner 数据仓库同时提供了直观简便的 OLAP 工具，以构建数据立方体，完成各种多维数据分析操作。数据的抽取和集成、数据的组织及 OLAP 的实现等均由元数据统一管理。

13.7 算法库管理

算法是数据挖掘的核心部件，数据挖掘任务是通过有机组织的算法库来实现的。首先这些算法是可操作的，系统通过元数据来记录算法的存储位置，输入输出参数及规则的表示形式；其次算法应该与系统平滑集成，我们采用动态链接库（Dynamical Link Library）的形式嵌入算法，这一方面增强了系统的可移植性，另一方面，也为系统节省了部分内存的开销（DLL 的动态装入与卸载）；最后算法的接口应该具有统一的标准，以便不同的采掘步骤能一致地调用算法。

13.7.1 数据挖掘算法的元数据

在 MSMiner 元数据中，各种数据挖掘算法的详细信息主要通过元数据类 CAlgorithm 来描述：

CAlgorithm

{

Attributes:

AlgorithmID	算法 ID
AlgorithmName	算法名称
AlgorithmType	算法类别（分类、预测、聚类等等）
AlgorithmFile	算法库 DLL 的路径
AlgorithmFunction	算法函数定义
AlgorithmParas	算法参数声明
Remark	算法的说明信息
SourceTableFormat	源数据表格式
TargetTableFormat	目标数据表格式
Evaluation	对算法结果的评价方法

```

... ..
Methods:
    Update()
... ..
}

```

其中，AlgorithmParas 是元数据集合类 CAlgorithmParaS 的对象实例，其中包含了元数据个体类 CAlgorithmPara 的对象实例的集合。

类 CAlgorithmPara 的定义如下：

```

CAlgorithmPara
{
Attributes:
    AlgorithmID        所属的算法 ID
    ParaID             参数 ID
    ParaName           参数名
    ParaType           参数数据类型
    ParaNo             参数序号
    DefaultValue       缺省值
    MaxValue           最大值
    MinValue           最小值
    ...
Methods:
    Update()
    ... ..
}

```

各个算法的元数据中包含对算法的功能、用途及使用方法等信息的详细说明。在建立数据挖掘任务模型时，任务向导将根据这些信息引导用户逐步选择合适的算法、设置各种参数，将算法的存储位置、函数接口定义、参数设置以及数据源设置等信息传递给外部调用函数，从而实现算法的调用。

13.7.2 可扩展性的实现

MSMiner 系统中各种数据挖掘核心算法以动态链接库 DLL 的形式实现，并在元数据中登记注册。采用 DLL 方式实现各数据挖掘算法有两个主要的优点：

- 以 DLL 可执行代码实现的算法在系统运行时动态载入，保证了算法的执行效率。
- DLL 算法库独立于 MSMiner 系统而存在，便于维护、升级和扩展，同时具有良好的通用性。

MSMiner 系统提供专门的算法管理模块来查看和维护各种算法的有关信息。用户还可以按照一定的规范，开发自己的算法 DLL，并在算法管理模块中进行注册，使数据挖掘任务能够使用新的算法，从而实现算法库的可扩展性，使整个数据挖掘系统具有很强的灵活性和通用性。

用户可在算法注册向导的指导下，逐步完成对新算法的动态加载。加载过程包括算法一般信息的注册和算法参数的注册。其流程图如图 13.11 所示：

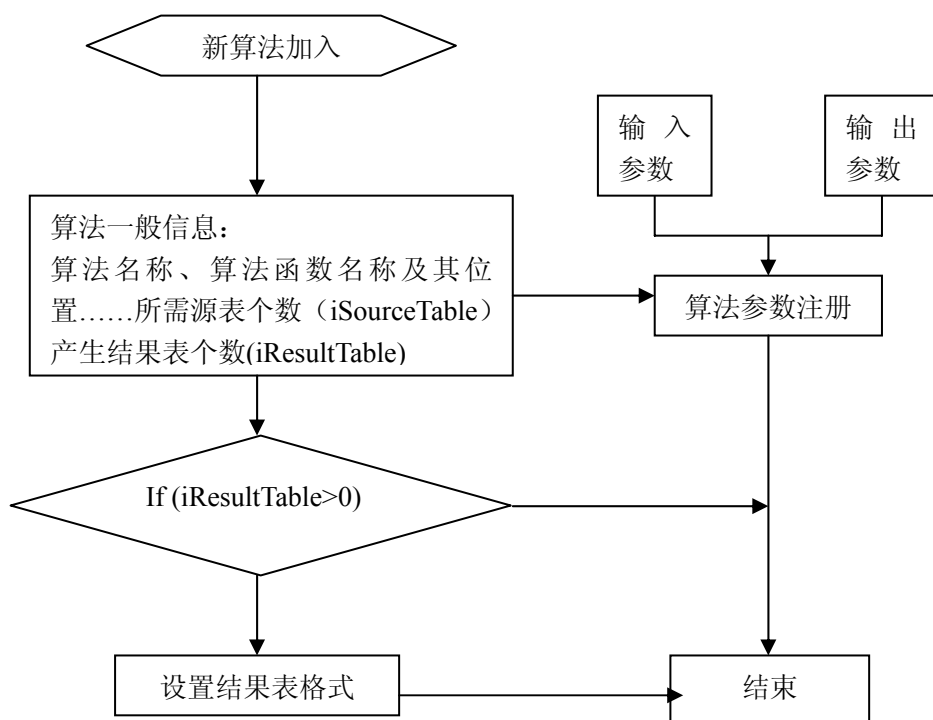


图 13.11 流程图

13.7.3 采掘算法的接口规范

为了提供尽可能全面的数据挖掘解决方案，我们将集成适用于不同任务的多种数据挖掘算法，但对于有些特定的数据挖掘任务，系统提供的算法将可能难以胜任，此时用户往往希望能够加入自己的处理方案。因此，系统必须提供一个开放的、灵活通用的接口，使用户能够加入新的算法。

这里我们给出约定，用户编写自己的算法，生成动态链接库 DLL 文件时，必须符合以下的规范：

1.挖掘算法调用参数的格式

每个数据挖掘任务由一个或几个采掘步骤组成，这些采掘步骤构成有向图。数据挖掘引擎负责拓扑排序，按顺序方向执行每一步骤。执行每一采掘步骤时，数据挖掘引擎分配相应的参数给采掘算法，由采掘算法执行相应的任务并返回相应的结果。采掘算法调用参数的格式统一定为（functionName 为调用函数名称）：

functionName(string MetadataCn, Variant stepID)

说明：metadataCn 是系统元数据的连接串，stepID 是该算法对应的采掘步骤的标示号。算法 functionName 通过这两个参数，访问元数据表 metaStepPara 获得他需要的参数。

2.MetaAlgorithmPara 的格式及记录组织。

ParaID (参数的标示号)	ParaName (参数名称)	FlagofPara (参数类别标示)	ParaType (参数的数据类型)	Para(int,float,string)Value (参数值)	StepID (参数对应的步骤标示号)
	SourceCN	001	String		
	SourceRS	011	String		

	SourceCN2	002	String	其它数据源链接参数	
	SourceRS2	012	String	其它数据源	
	
	ResultCN	101	String	可以缺省，缺省时与 001 SourceCN 相同	
	ResultRS	111	String	输出结果表表名	
	FormatOfResult	121	String	输出结果样本表的表名	
		
	inputPara	201	不定		
	outputPara	301	不定		

说明：

参数类别编号说明：第一位是特征位： “0” ----源数据信息，
“1” ----结论数据信息（规则等）
“2” ---精度参数（算法开始前设定）
“3” ---评估参数（算法执行返回前设定）
第二位是关联位： “0” --- 连接串
“1” ---结果表表名
（第一位是“2，3”此位无意义）
“2” ---输出结果样本表的表名
第三位是序号位： 表示此类别参数的个数

3.算法需提供以下信息，以便在系统中注册其参数信息。

- 算法文件名称 (FileName)： 算法所在的 DLL 文件的全称（含路径名）
- 采掘算法调用参数的格式函数名称 (FunctionName)： 执行采掘算法的函数名称。
- 函数序号(FunctionNo)： 函数在 DLL 输出函数中的序号。
- 函数功能说明(Remark)：
- 所需源表个数 (NumberOfSourceData)： 需要几个表作为数据源。
- 结果表个数 (NumberOfResult)： 算法的结果放在几个表中。
- 结果表的样式/样板 (FormatOfResult)： 预先定义的一种空表，用来生成用来存放结果表的结构。
- 输入参数 (inputPara)： 采掘算法所要求的参数，如可信度，支持度，误差等；
- 输出参数 (outputPara)： 采掘算法所返回的简单结果，采用输出参数的方式予以保存。例如，在决策树的测试算法中，评估的结果，如预测精度，决策树包括的节点数等简单的结果。

采掘算法通过传递的参数，访问元数据的算法参数信息表，获得所需要的输入参数和数
据，算法运行完毕后将输出的评价参数和规则写到相应的表中。通过这个接口系统可以方便
的注册算法的参数信息，使采掘算法与采掘任务有机结合起来。

MSMiner 通过元数据记录采掘算法的描述信息，使得采掘算法像数据库中的数据一样易

于操作；采用动态链接库的形式嵌入采掘算法，采用统一的算法接口规范算法，实现了采掘算法的动态加载。目前 MSMiner 算法库中包含决策树、BP 及 SOM 神经网络、多元回归分析及关联规则发现算法、粗糙集属性约简算法、支持向量机算法、模糊据类算法等，可用于分类、预测、关联规则发现等数据挖掘任务。

习 题

1. 解释知识发现的概念。
2. 知识发现过程主要包括那几部分内容，它们之间的关系是什么。
3. 简述知识发现中包括哪些任务。
4. 什么是分类，什么是聚类，简述二者的区别和联系。
5. 简述数据仓库的定义，讨论数据仓库与传统意义上的关系型数据库之间的异同。
6. 假设数据仓库由 4 维构成，即时间、项、部门、地点， 并有两个测度：即销售总额（dollars_sold）、 销售单位（units_sold）。 请画出销售的数据仓库星型模式。
7. 简述元数据的概念，以及数据仓库中元数据管理策略。
8. 描述 ETL 任务的具体内容，为什么数据仓库中需要 ETL 工作。
9. 解释维度、数据立方体、度量的概念，以及 OLAP 中的星型模型和雪花模型。
10. 比较 OLAP、OLTP 的概念，以及 ROLAP 和 MOLAP 的概念。
11. 解释 OLAP 术语中的渐变维度（Slowly Changing Dimension）概念。

第十四章 分布智能

分布智能主要研究在逻辑上或物理上分散的智能系统如何并行地、相互协作地实现问题求解。主体计算(agent computing)不仅是分布智能的研究热点,而且可能成为下一代软件开发的重要突破点。

14.1 概述

人类活动大部分都涉及社会群体,大型复杂问题的求解需要多个专业人员或组织协作完成。自 20 世纪 70 年代后期以来,随着计算机网络、计算机通信和并行程序设计技术的发展,分布智能的研究逐渐成为一个新的研究热点。20 世纪 90 年代以来,互联网的迅速发展为新的信息系统、决策系统和知识系统的发展提供极好的条件,他们在规模、范围和复杂程度上增加极快,分布智能技术的开发与应用越来越成为这些系统成功的关键。分布智能系统具有下列特色:

(1) 系统中的数据、知识,以及控制不但在逻辑上,而且在物理上是分布的,既没有全局控制,也没有全局的数据存储。

(2) 各个求解机构由计算机网络互连,在问题求解过程中,通信代价要比求解问题的代价低得多。

(3) 系统中诸机构能够相互协作,来求解单个机构难以解决,甚至不能解决的任务。

分布式人工智能系统的实现可以克服原有专家系统、学习系统等的弱点,极大地提高知识系统的性能,其主要优点表现为:

(1) 提高问题求解能力。由于分布的特点,分布智能系统的问题求解能力大大提高。首先可靠性高,通信路径、处理结点、以及知识的冗余使得出现故障时,整个系统仅仅降低响应时间或求解精度,而不至于完全瘫痪;其次,系统容易扩展,增加处理单元可以扩大系统的规模并且提高问题求解能力;再者就是系统的模块性,将使整个系统设计十分灵活。

(2) 提高问题求解效率。由于分布智能系统中各结点可以并行地求解问题,所以可以开发问题求解中的并行性,提高求解效率。

(3) 扩大应用范围。分布智能技术可以打破目前知识工程领域的一个限制,即仅仅使用一个专家。在分布智能系统中,不同领域、甚至同一领域的不同专家可以协作求解某一专家不能解决或不能很好解决的问题。同时,许多领域中若干非专家有机地结合起来,也可能达到或超过一个专家的水平。

(4) 降低软件的复杂性。分布智能系统将整个求解任务分解成若干相对独立的专门的子任务,其结果是降低了各个处理结点问题求解的复杂性。

分布智能的研究可以追溯到 70 年代末期。早期分布智能的研究主要是分布式问题求解,其目标是要创建大粒度的协作群体,它们之间共同工作以对某一问题进行求解。在一个纯粹的 DPS 系统中,问题被分解成任务,并且为求解这些任务,需要仅为该问题设计一些专用的任务执行系统。所有的交互策略都被集成为系统设计的整体部分。这是一种自顶向下设计的系统,因为处理系统是为满足在顶部所给定的需求而设计的。Hewitt 和他的同事们研制了基于 ACTOR 模型的并发程序设计系统[Hewitt 1983]。ACTOR 模型提供了分布式系统中并行计算理论和一组专家或 ACTOR 获得智能行为的能力。在 1991 年 Hewitt 提出开放信息系统语义

[Hewitt 1991], 指出竞争、承诺、协作、协商等性质应作为分布式人工智能的科学基础, 试图为分布式人工智能的理论研究提供新的基础。1980 年 Davis 和 Smith 提出了合同网 (CNET) [Smith 1980]。CNET 使用投标—合同方式实现任务在多个节点上的分配。合同网系统的重要贡献在于提出了通过相互选择和达成协议的协商过程实现分布式任务分配和控制的思想。分布式车辆监控测试系统 DVMT 也是分布式人工智能领域最早和最有影响的研究课题之一, 由麻萨诸塞大学的 Lesser, Corkill 和 Durfee 等人主持研制 [Lesser 1980]。该系统对市区内行驶的车辆轨迹进行监控, 并以此环境为基础, 对分布式问题求解系统中许多技术问题进行研究 [Durfee 1987]。DVMT 是以分布式传感网络数据解释为背景, 对复杂的黑板问题求解系统之间的相互作用进行了研究, 提供了抽象和模型化分布式系统行为的方法。Gasser 等人于 1987 年研制了 MACE 系统, 是一个实验型的分布式人工智能系统开发环境 [Gasser 1987]。MACE 中每一个计算单元都称作主体, 它们具有知识表示和推理能力, 主体之间通过消息传送进行通信。MACE 是一个类面向对象环境, 但避开了并发对象系统中难于理解和实现的继承问题。MACE 的各个机构并行计算, 并提供了描述机构的描述语言, 具有跟踪的 demons 机制。该课题研究的重点是在实际并行环境下运行分布智能系统, 保持概念的清晰性。中国科学院计算技术研究所史忠植等研究了分布式知识处理系统 DKPS [朱建立 1990]。该系统采用逻辑——对象知识模型, 研究了知识共享和协作求解等问题。基于 ACTOR 计算模型, Ferber 等人于 1991 年研制了 Mering IV 反射 ACTOR 语言 [Ferber 1991]。在这个模型中 actor 是可以并发执行的活动对象, 通过异步消息传送实现相互作用。Mering IV 是反射语言, 可以在结构上和操作上表示自身。系统利用反射性使得不同大小和粒度的智能主体能够以统一的方式相互作用。

90 年代, 多主体系统 (multiagent system, MAS) 的研究成为分布智能研究的热点。多主体系统主要研究自主的智能主体之间智能行为的协调, 为了一个共同的全局目标, 也可能是关于各自的不同目标, 共享有关问题和求解方法的知识, 协作进行问题求解。基于主体的概念, 人们提出了一种新的人工智能定义: “人工智能是计算机科学的一个分支, 它的目标是构造能表现出一定智能行为的主体”。所以, 智能主体的研究应该是人工智能的核心问题。斯坦福大学计算机科学系的 Hayes-Roth 在 IJCAI'95 的特邀报告中谈到: “智能的计算机主体既是人工智能最初的目标, 也是人工智能最终的目标。” 现在, 关于主体的研究不仅受到了人工智能研究人员的关注, 也吸引了数据通信、人机界面设计、机器人、并行工程等各领域研究人员的兴趣。

14.2 分布式问题求解

分布式问题求解是分布智能研究的一个分支, 由于人们认识到解决一类复杂问题和综合性问题时集中式解题系统的局限性, 加之网络技术、通信技术和并行程序设计技术取得了重要进展, 这方面的研究逐渐受到重视。

在分布式问题求解系统中, 数据、知识、控制均分布在系统的各结点上, 既无全局控制, 也无全局数据和知识存贮。由于系统中没有一个结点拥有足够的数据和知识来求解整个问题, 因此各结点需要交换部分数据、知识、问题求解状态等信息, 相互协作来进行复杂问题的协作求解。

分布式问题求解系统有两种协作方式, 即任务分担 (task sharing) 和结果共享 (results sharing)。Smith 和 Davis 提出了任务分担方式。在任务分担系统中, 结点之间通过分担执行整个任务的子任务而相互协作, 系统中的控制以目标为指导, 各结点的处理目标是为了求解整个任务的一部分。Lesser 和 Corkill 提出了结果共享方式。在结果共享方式的系统中, 各结点通过共享部分结果相互协作, 系统中的控制以数据为指导, 各结点在任

何时刻进行的求解取决于当时它本身拥有或从其它结点收到的数据和知识。

任务分担的问题求解方式适合于求解具有层次结构的任务,如工厂联合体生产规划、数字逻辑电路设计、医疗诊断。结果共享的求解方式适合于求解与任务有关的各子任务的结果相互影响,并且部分结果需要综合才能得出问题解的领域。如分布式运输调度系统、分布式车辆监控实验系统 DVMT。事实上,任务分担与结果共享两种求解方式,只是强调问题求解的阶段不同,它们本质上并非是不相容的。Smith 从任务分担的角度研究了两种方式结合的方法, Hayes-Roth 从结果共享的角度研究了两种方式的结合方法。

14.2.1 分布式问题求解系统分类

分布式求解系统中的组织结构是指结点之间的信息与控制关系以及问题求解能力在结点中的分布模式,它说明了各结点的作用和结点间的关系。这种组织结构必须有利于减少有关协作的不确定性。

根据组织结构,分布式问题求解系统可以分为三类,即层次结构类、平行结构类、混合结构类,其特点分别简述如下:

(1) 层次结构类

- 从组成结构上看,任务是分层的。每个任务由若干个子任务组成,每个子任务又由若干个下层子任务组成,以此类推,形成层次结构。

- 各个子任务在逻辑上或物理上是分布的。

层次结构类任务可采用任务分担模型进行求解,每个处理结点求解一个或几个任务。如分布式运输调度系统、分布式车辆监控实验系统 DVMT 等,均属于这种类型。

(2) 平行结构类

- 从组成结构上看,任务是平行的。每个任务由若干子任务组成,各子任务性质相同,具有平行关系。

- 整个任务的完成依赖于与之有关的各个子任务,各子任务的部分解往往是不完全的,需要综合成整个任务的解。

- 各个子任务在时间或空间上往往是分布的。

平行结构类任务可采用结果共享模型进行求解,每个处理结点求解一个或几个任务。Kornfeld 和 Hewitt 根据“科学团体是高度并行的系统”的观点,提出了名为“科学团体例子”的平行结构。

(3) 混合结构类

- 从组成结构上看,任务是分层次,而每层中的任务是并行的。

- 各个子任务是分布的。

混合结构类任务可以采用任务分担与结果共享相结合的问题求解模型。

14.2.2 分布式问题求解过程

分布式问题求解系统的求解过程可以分为四步:任务分解,任务分配,子问题求解,以及结果综合。系统首先从用户接口接收用户提出的任务,判断是否可以接受。若可以接受,则交给任务分解器,否则通知用户该系统不能完成此任务。任务分解器将接收的任务,按一定的算法分解为若干相对独立、但又相互联系的子任务。若有多个分解方案,则选出一个最佳方案交给任务分配器。任务分配器将接受的任务,按一定的任务分配算法,将各子任务分配到合适的结点。若有多个分布方案,则选出最佳方案。各求解器在接受到子任务后,与通信

系统密切配合进行协作求解，并将局部解通知协作求解系统，之后该系统将局部解综合成一个统一的解并提交用户。若用户对结果满意，则输出结果，否则再将任务交给系统重新求解。事实上，实际系统中的问题求解要比上述过程复杂得多。

下面简单介绍几种典型的任务分解和任务分配的方法：

(1) 合同网络。Davis 和 Smith 提出的“合同网络”本质上是一种适合任务分担求解系统的任务分配算法。在此方法中，合同就是产生任务的结点与愿意执行此任务的结点之间达成的一种协议，建立合同的思想与“工业招标”类似。Davis 和 Smith 以分布式感应系统为背景实现了合同网络。

(2) 动态层次控制。Findler 等在合同网络的基础上，提出动态层次控制的任务分解与任务分配方法。按此方法，系统在任务分解之后，首先对各结点的处理能力进行分析，综合问题求解环境数据，建立结点问题控制关系框架与全局性冲突监控关系网络，然后按控制关系框架分布任务进行协作求解。求解过程中出现的条件资源冲突分别通过商议、启发式知识，以及全局性冲突监控网络来解决。Findler 等在工厂联合体生产规划系统对上述思想做了模拟实验。

(3) 自然分解，固定分配。Lesser 和 Corkill 等在分布式车辆监控实验系统 DVMT 中采用了“自然分解，固定分配”的任务分解与任务分布算法。据此方法，预先将被监控区域划分为若干相互重叠的子区域，各子区域内的传感器将收到的数据送至邻近结点进行处理。

(4) 部分全局规划。Durfee 和 Lesser 提出了一种灵活的协调方式，即部分全局规划。一个部分全局规划包括目标信息、规划活动图、解结果构造图，以及状态信息四部分。目标信息包括部分全局规划的最终目标和重要性等信息；规划活动图表示结点的工作，如结点目前正进行的主要规划步骤及其成本、期望结果等；解结果构造图中的信息说明结点之间的交互关系；状态信息记录自其它结点收到的有关信息的指针、收到时间等。通过交换部分全局规划，可以进行动态的任务分解与任务分布。

以上四种方法中，前两种适合于任务分担方式求解的系统，后两种适合于结果共享方式求解的系统。

协作是分布式问题求解研究的重要方面。根据结点间协作量的多少，将分布式问题求解系统中的协作分为三类：

(1) 全协作系统。在这类系统中，结点可以为了系统目标或其它结点的目标而对本身的求解目标进行修改，从而能处理相互依赖的子问题，协作量大，且通信成本高。

(2) 无协作系统。各结点根本不协作，无通信成本。

(3) 半协作系统。在一定程度上进行协作的系统。

在分布式问题求解系统中，保持全局一致性意味着各结点的问题求解朝着有利于实现系统目标的方向进行。Corkill, Lesser 和 Durfee 研究了分布式问题求解系统中通过组织结构实现元级控制的方法。Naseem 和 Pamesh 研究了基于知识的分布式系统中不确定处理的问题，并给出了一个结果综合模型。

在分布式问题求解中，常用的通信方式有共享全局存储器、信息传递，以及二者的结合。黑板模型是分布式问题求解系统中使用较多的框架结构。Lesser 等人在 DVMT 系统的黑板模型中，增加了目标黑板、抽象级黑板以及复杂的规划机制。系统中聚合器根据数据黑板上的信息产生规划所用的抽象数据，并将其存放在抽象级黑板上。目标处理器负责目标的分解，并产生知识源例示队列。规划程序结合抽象级数据、知识源例示队列以及网络系统的组织结构信息进行处理，产生规划队列。系统运行时根据本体黑板信息的变化状态以及网络中其它求解器的状态，对规划的实施进行动态调整。

主体结构需要解决的问题是主体由哪些模块组成，它们之间如何交互信息，主体感知到的信息如何影响它的行为和内部状态，以及如何将这些模块用软件或硬件的方式组合起来形

成一个有机的整体，真正实现主体。

14.3 主体基础

多主体系统主要研究在逻辑上或物理上分离的多个主体协调其智能行为，即知识、目标、意图及规划等，实现问题求解。多主体系统可看作是一种由底向上设计的系统，因为在原理上，分散自主的主体首先被定义，然后研究怎样完成单个或几个主体的任务求解。多主体系统不仅可以处理单一目标，也可以处理不同的多个目标。

主体相应的英文词是“Agent”。在英文中“Agent”这个词主要有三种涵义：一是指能对其行为负责的人；二是指能够产生某种效果的，在物理、化学或生物意义上活跃的东西；三是指代理，即接收某人的委托并代表他执行某种功能，因此有人称为代理。在计算机和人工智能领域中，主体可以看作是一个实体，它通过传感器感知环境，通过效应器作用于环境。若主体是人，则传感器有眼睛、耳朵和其它器官，手、腿、嘴和身体的其它部分是效应器。若主体是机器人，摄像机等是传感器，各种运动部件是效应器。一般主体可以用图 14.1 表示。

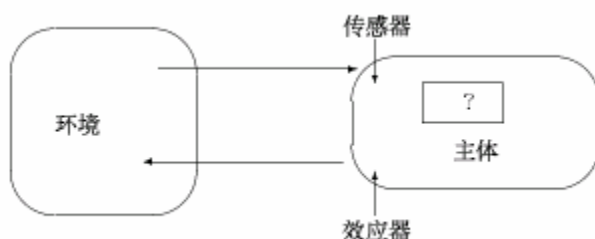


图 14.1 主体与环境交互作用

通常认为一个主体需要具有下述部分或全部特性：

(1) 自治性。这是一个主体的基本特性，即可以控制它自身的行为。主体的自治性体现在：主体的行为应该是主动的、自发的；主体应该有它自己的目标或意图（intention）；根据目标、环境等的要求，主体应该对自己的短期行为作出计划。

(2) 交互性，即对环境的感知和影响。无论主体生存在现实的世界中（如机器人、Internet 上的服务主体等）还是虚拟的世界中（如虚拟商场中的主体等），它们都应该可以感知它们所处的环境，并通过行为改变环境。一个不能对环境作出影响的物体不能被称为主体。

(3) 协作性。通常主体不是单独地存在，而是生存在一个有很多个主体的世界中。主体之间的良好有效的协作可以大大提高整个多主体系统的性能。

(4) 可通信性。这也是一个主体的基本特性。所谓通信，指主体之间可以进行信息交换。更进一步，主体应该可以和人进行一定意义下的“会话”。任务的承接、多主体的协作、协商等都以通信为基础。

(5) 长寿性（或时间连贯性）。传统程序由用户在需要时激活，不需要时或者运算结束后停止。主体与之不同，它应该至少在“相当长”的时间内连续地运行。这虽然不是主体的必须特性，但目前一般认为它是主体的重要性质。

另外，有些学者还提出主体应该具有自适应性、个性等特性。在实际的应用中，主体经常需要在时间和资源受到一定限制的情况下作出行动。所以，对于现实世界中的主体，除了应该具有主体的一般性质外，还应该具有实时性。

目前，多主体系统的研究非常活跃。多主体系统试图用主体来模拟人的理性行为，主要应用在对现实世界和社会的模拟、机器人和智能机械等领域。而在现实世界中生存、工作的主体，要面对的是一个不断变化的环境。在这样的环境中，主体不仅要保持对紧急情况的及

时反应，还要使用一定的策略对中短期的行为作出规划，进而通过对世界和其它主体的建模分析来预测未来的状态，以及通过通讯语言实现和其他主体的协作或协商。

14.4 主体理论

14.4.1 理性主体

1987 年 Bratman 从哲学上对行为意图(intention)的研究对人工智能产生了广泛的影响。他认为只有保持信念(belief)、愿望(desire)和意图(intention)的理性平衡，才能有效地解决问题。在开放世界中，理性主体(rational agent)的行为不能直接由信念、愿望，以及由两者组成的规划驱动，在愿望与规划之间应有一个基于信念的意图存在。其原因是：

(1) 主体的行为受有限资源的约束，一旦主体决定做什么，就建立了一个承诺(commitment)的有限形式。

(2) 在多主体环境中，需要由承诺来协调各主体的行为。若无承诺，则无从谈行为。意图正是一种承诺的选择。

在开放和分布的环境中，一个理性主体的行为受制于意图。意图又表现为：

(1) 一个主体要改变自己已有的意图必须要有理由；

(2) 一个主体不能无视环境的变化而坚持不符合实际或已不重要的意图。

理性平衡的目的在于使理性主体的行为符合环境的特性。所谓环境特性不仅仅指环境的客观条件，同时包含环境中的社会团体因素，如社会团体关于理性行为的判断法则。Bratman 给出了意图—行为原则：如果主体 A 有进行行为 B 的当前行为意图是合理的，那么 A 把意图转化为行为，有意地进行行为 B 就是合理的。

在给定时间里，主体理性表现为：

(1) 性能测度规定成功的程度。

(2) 主体感知所有事情，我们将把这个完整的感知历史称为感知序列。

(3) 主体知道环境是什么。

(4) 主体可以执行的动作。

一种理想的理性主体可以定义为：对于每种可能的感知序列，理想的理性主体，在感知序列所提供的证据和主体内部知识的基础上，应该做的所期望的动作是使它的性能测度为最大。

14.4.2 BDI 主体模型

BDI 主体模型可以通过下列要素描述：

(1) 一组关于世界的信念；

(2) 主体当前打算达到的一组目标；

(3) 一个规划库，描述怎样达到目标和怎样改变信念；

(4) 一个意图结构，描述主体当前怎样达到它的目标和改变信念。

Rao 和 Georgeff 提出了一个简单的 BDI 解释器(Rao and Georgeff 1992)：

BDI-Interpreter

initialize-state();

do

```

options := option-generator(event-queue, B, G, I);
selected-options := deliberate(options, B, G, I);
update-intentions(selected-options, I);
execute(I);
get-new-external-events();
drop-successful-attitudes(B, G, I);
drop-impossible-attitudes(B, G, I);
until quit

```

14.4 主体结构

14.4.1 主体基本结构

主体可以看成是一个黑箱，通过传感器感知环境，通过效应器作用环境。人主体的传感器有眼、耳、鼻、以及其它器官，而手、腿、嘴、身体可以看成效应器。机器人主体有摄像机等作为传感器，而各种马达是效应器。软件主体通过字符串编码作为感知和作用（见图 14.2）。

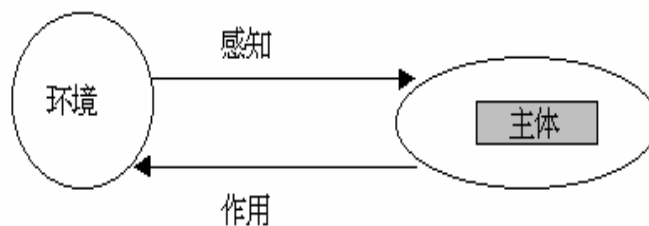


图 14.2 黑箱软件主体

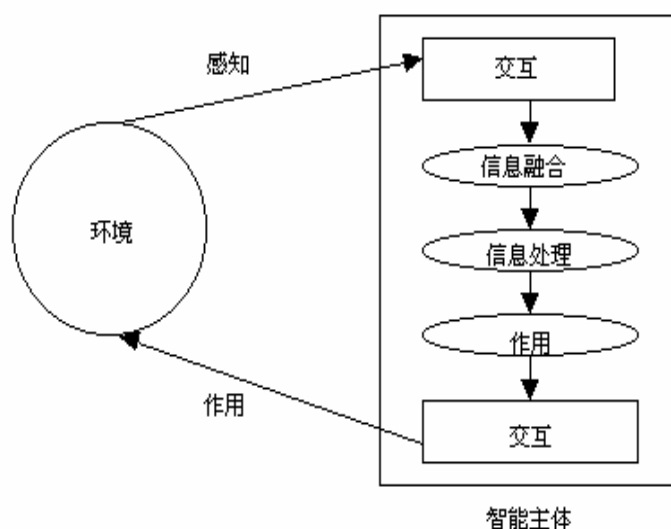


图 14.3 智能主体的工作过程

大多数主体不仅要与环境交互作用，更主要的是处理和解释接收的信息，达到自己的目的。图 14.3 给出了智能主体的工作过程。主体接收到的信息首先要以适当的方式进行融合，

并能为主体知识库所接受。信息融合特别重要，不同交互模块得到的结果可能不同，表达方式也不一样。例如，对于同一件事，人提供的信息与智能主体得到的内容和形式不相同。信息融合要识别和正确区分这种不一致性。

一旦主体接收外部信息，信息处理过程成为主体的核心，因为它反应主体的真正功能。信息处理的目的是解释可用的数据，形成具体规划。因为每个主体都有具体的目标，内部目标的影响必须作为影响的一部分考虑。如果影响弄清楚了就要采取行动，使之达到或接近目标。形成规划时主体可以规定知识，包括对新情况反应的具体处理步骤。但是，这不是本质的东西，因为主体执行可以不要规划。当要求对环境对象交互时，动作模块将使用合适的交互模块。控制执行也是动作模块的任务。

由此可见，主体可以定义为从感知序列到主体例示动作的映射。设 \mathbf{O} 是主体随时能注意到的感知集合， \mathbf{A} 是主体在外部世界能完成的可能动作集合，则主体函数 $f: \mathbf{O}^* \rightarrow \mathbf{A}$ 定义在所有环境下主体的行为。人工智能的任务是设计主体程序，实现从感知到动作的映射。一个主体骨架程序给出如下：

```
function Skeleton-Agent(percept) return action
    static: memory /* 主体的世界记忆 */
    memory  $\leftarrow$  Update-Memory(memory, percept)
    action  $\leftarrow$  Choose-Best-Action(memory)
    memory  $\leftarrow$  Update-Memory(memory, action)
    return action
```

每次调用，主体的记忆将修改，反应新的感知。理想的理性主体对于每一个可能的感知序列，总希望达到最好的性能。因此，这里主体采取最佳动作。所采取的动作也保存在记忆中。

不是所有的主体动作都表达为对新的情况的反应，主体也可以创建新的规划。在这种情况下，信息提供者的知识只是在特定时间有用。这种知识直接导致慎思主体和反应主体的重要区别。

14.4.2 慎思主体

慎思主体 (Deliberative agent)，也称作认知主体 (Cognitive agent)，是一个显式的符号模型，包括环境和智能行为的逻辑推理能力。它保持了经典人工智能的传统，是一种基于知识的系统 (Knowledge-based system)。环境模型一般是预先实现的，形成主要部件知识库。采用这种结构的主体要面对以下两个基本问题：

转换问题：如何在一定的时间内将现实世界翻译成一个准确的、合适的符号描述；表示/推理问题：如何用符号表示复杂的现实世界中的实体和过程，以及如何让主体在一定的时间内根据这些信息进行推理作出决策。

第一个问题导致了计算机视觉、自然语言理解等多个领域的研究；第二个问题导致了知识表示、自动推理、自动规划等多个领域的研究。在实现中这些都有一定的难度。因为表示太复杂，慎思主体适应动态环境有一定程度的局限性。没有必要的知识和资源，在主体执行时要加入有关环境的新信息和知识到它们已有的模型中是困难的。

慎思主体是具有内部状态的主动软件，它与具体的领域知识不同，具有知识表示、问题求解表示、环境表示、具体通信协议等。根据主体思维方式的不同，慎思主体可以分为抽象思维主体、形象思维主体。抽象思维主体是基于抽象概念，通过符号信息处理进行思维。形象思维主体是通过形象材料进行整体直觉思维，与神经机制的连接论相适应。

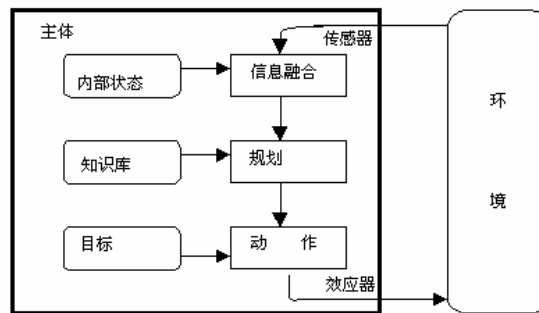


图 14.4 慎思主体的框图

图 14.4 给出了慎思主体的框图。主体通过传感器接收外界环境的信息，根据内部状态进行信息融合，产生修改当前状态的描述。然后，在知识库的支持下制定规划。形成一系列动作，通过效应器对环境发生作用。慎思主体程序如下：

```
function Deliberate-Agent(percept) returns action
static: environment, /* 描述当前世界环境 */
       kb, /* 知识库 */
       plan /* 规划 */

environment ← Update-World-Model(environment, percept)
state ← Update-Mental-State(environment, state)
plan ← Decision-Making(state, kb, action)
environment ← Update-World-Model(environment, action)
return action
```

上述程序中，Update-World-Model 函数从感知产生当前世界环境的抽象描述。Update-Mental-State 函数根据当前感知到的环境，修改主体内部的心智状态。知识库包括通用的知识和实际的知识。主体运用知识，通过 Decision-Making 函数可以进行决策，制定规划。主体执行所选的动作，并通过 Update-World-Model 函数与环境发生交互。

一个典型的慎思主体是 BDI 结构（见图 14.5）。BDI 主体模型可以通过下列要素描述：① 一组关于世界的信念；② 主体当前打算达到的一组目标；③ 一个规划库，描述怎样达到目标和怎样改变信念；④ 一个意图结构，描述主体当前怎样达到它的目标和改变信念。

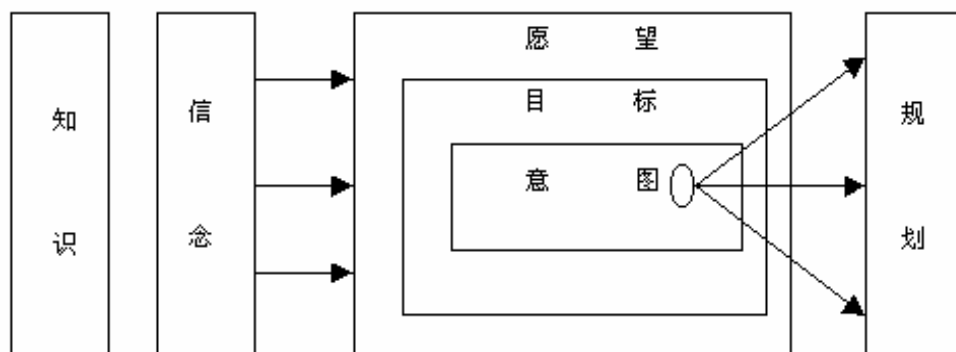


图 14.5 BDI 结构

根据 BDI 结构，Rao 和 Georgeff 提出了一个简单的 BDI 解释器[Rao 1992]。

```
BDI-Interpreter () {
  initialize-state();
  do
```

```

options := option-generator(event-queue, B, G, I);
selected-options := deliberate(options, B, G, I);
update-intentions(selected-options, I);
execute(I);
get-new-external-events();
drop-successful-attitudes(B, G, I);
drop-impossible-attitudes(B, G, I);
until quit
}

```

BDI 模型影响慎思主体的系统结构。1988 年, Bratman 等研制了 IRMA (Intelligent Resource-Bounded Machine Architecture) 系统[Bratman 1987]。图 14.6 给出了 IRMA 主体结构。在 IRMA 结构中, 一个主体有四个关键的符号数据结构, 分别为一个规划库 (plan library), 用符号表示的信念 (belief)、愿望 (desire) 和意图 (intention)。另外, 每个主体有一个推理机, 用来对世界进行推理; 有一个分析机, 以决定哪个规划可以用来完成该主体的意图; 有一个“机遇分析机”, 监视环境的变化; 有一个过滤处理器 (filtering process); 还有一个慎思处理器 (deliberation process)。过滤处理器决定主体将要进行的动作序列是否和该主体当前的意图协调。慎思处理器在冲突的可选动作中作出选择。这种 IRMA 结构曾经在 “Tileworld” 这样的实验场景中进行过实验。

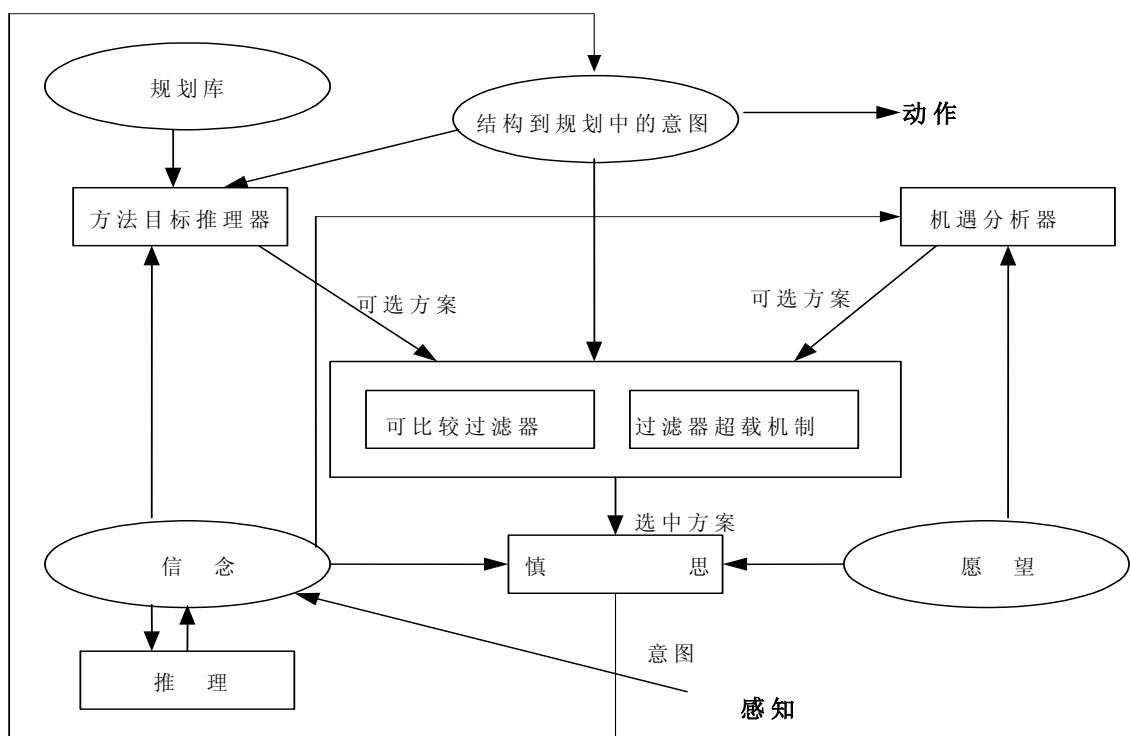


图 14.6 IRMA 主体结构

和 IRMA 结构相比, GRATE*更多地考虑到多主体系统的协作特性[Jennings 1993]。它采用一种层次结构 (见图 14.7)。同样, 主体的行为是由它的心智状态如信念、愿望、意图和联合意图等决定的。主体被分为两个部分, 一个是领域层, 一个是协作控制层。领域层用来解决领域问题, 例如工业控制、财政或交通等具体问题。协作控制层则控制领域层, 使它和其它主体的领域层能协调运作。协作控制层由三个通用的模块构成: 一个控制模块提供和领域层的接口; 一个情景评价模块; 还有一个协作模块。这里情景评价模块和协作模块都是用来使主体不仅能自己解决问题, 还可以和其它主体一起进行协作问题求解。采用 GRATE*结构的主体

社团曾在电力传输管理系统中做过实验。

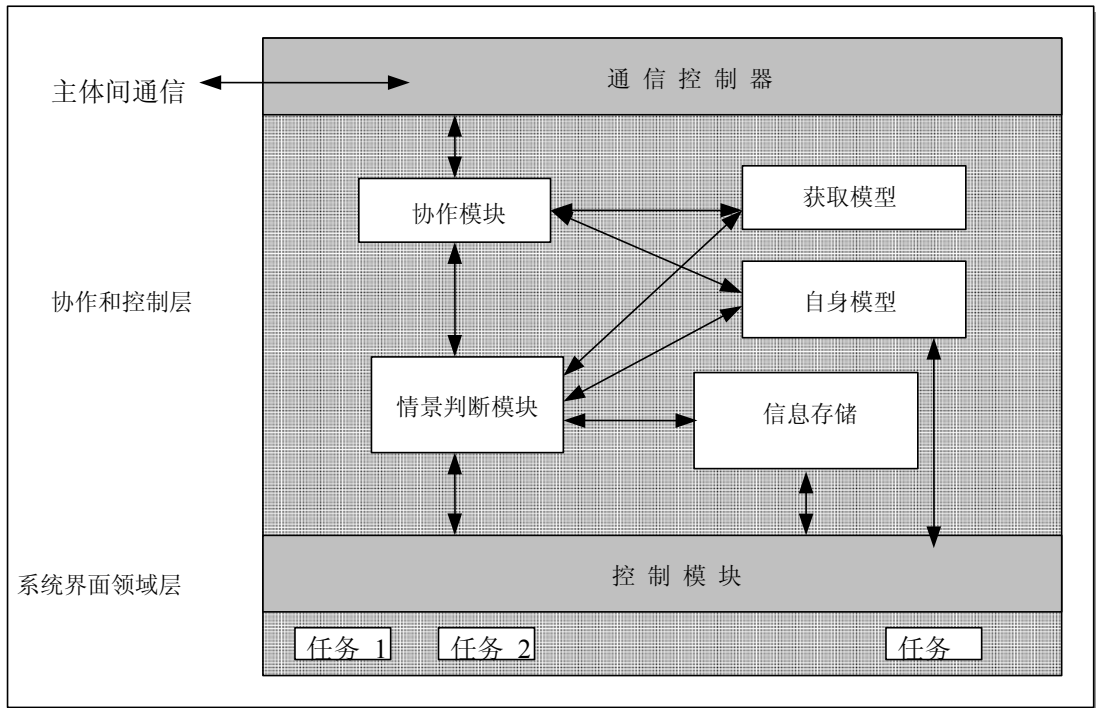


图 14.7 GRATE*主体结构

14.4.3 反应主体

传统人工智能的问题几乎没有改变地反映在慎思主体中。主要批评在于僵硬的结构。主体工作在非常动态变化的环境中，因此，它们必须有能力基于当前情景作出决策。但是，它们的意图和规划是根据过去特定时间的符号模型开发的，很少有变化。基于规划的僵硬结构相对扩大了弱点。因为规划器、调度器、执行器之间转换是很费时间的。当执行规划时情况已经或多或少地发生了变化。慎思主体的符号算法一般为理想的、可证明的结果设计的，经常导致高的复杂度。在动态环境下，对于相关情景在满足质量标准的前提下要求快速反应，比规划优化更重要。相反，在规划过程有效的情况下，慎思主体擅长进行规划的数学证明。

与此不同，反应主体（Reactive agent）是不包含用符号表示的世界模型，并且不使用复杂的符号推理的主体[Wooldridge 1995]。图 14.8 给出了反应主体的框图。图中条件——动作规则使主体将感知与动作连接起来。其中方块表示主体决策过程的当前的内部状态。椭圆表示过程中所用的背景信息。反应主体程序如下：

```
function Reactive-Agent(percept) returns action
static: state, /* 描述当前世界状态 */
        rules, /* 一组条件-动作规则 */
state ← Interpret-Input(percept)
rule ← Rule-Match(state, rules)
action ← Rule-Action[rule]
return action
```

上述程序中，Interpret-Input 函数从感知产生当前状态的抽象描述，Rule-Match 函数返回与给定状态描述匹配的规则组中的一条规则。

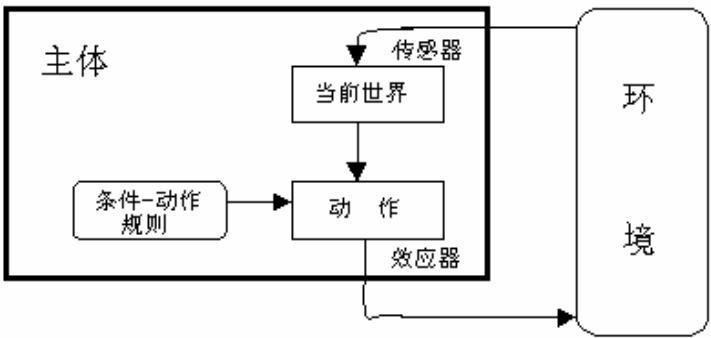


图 14.8 反应主体的框图

反应主体结构显然与 MIT 的 Brooks 所倡导的基于行为的人工智能有关 [Brooks 1991]。他认为智能行为是主体和它周围的环境交互的结果。Brooks 不仅是一个批评家，也是一个实践家。他实现了一些不使用符号表示和推理的机器人。这些机器人是基于一种“包含系统结构 (subsumption architecture)”。包含系统结构是一些层次状的能完成任务的行为。每个行为都试图控制机器人，因此彼此之间要竞争。底层的行为表示比较原始的行为（例如避开障碍物），它比高层的行为有更多的优先权。结果，用这种结构实现的系统相当简单，因为它不需要符号表示和推理。虽然简单，Brooks 表示它们可以完成符号人工智能系统能完成的任务。

就在 Brooks 提出基于行为的人工智能观点的同时，Chapman 正在完成他的硕士论文。在论文中，他也从理论上分析了符号人工智能的困难，并得到符号人工智能模型不适用于实际问题的结论。他和 Agre 开始寻找其它的方法。Agre 注意到，日常生活中的大部分活动都是“常规的 (routine)”。这些活动很少需要（甚至根本就不需要）重新进行抽象推理。很多任务一旦学会以后，就可以用一种常规的方法完成而几乎不需要变化。Agre 提出一个有效的主体结构可以基于“运行中项 (running argument)”的思想。就是说，既然大部分决策都是常规的，那么可以把它们编码成一种低级的结构（例如数字电路）。只要在一一定的时期（例如为了解决一个新的问题）更新这个低级结构就可以了。他们用这种思想实现了一个 PENG1 系统 [Agre 1987]。PENG1 是一个早期的计算机游戏的名字，玩家控制一个卡通人物在一个二维的场景中逃避敌人的追捕并设法杀死敌人。PENG1 系统就是用他们提出的结构实现的主体控制玩家的人物与敌人斗智。图 14.9 给出反应主体的结构。

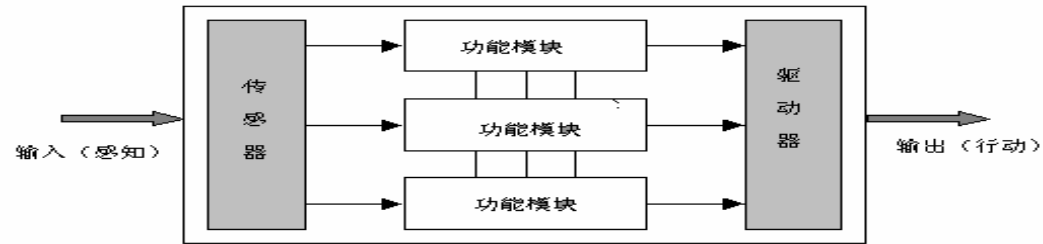


图 14.9 反应主体的结构

类似的系统还有 Maes 的“Agent network”结构 [Maes 1989]，以及 Steels 的“Mars

explorer” 系统[Stone 1999]等。

14.4.4 混合结构

前面我们讨论了主体的慎思结构和反应结构，它们反映了传统人工智能和基于行为的人工智能的特点。无论是纯粹的慎思结构还是纯粹的反应结构都不是构造主体的最佳方式。人们提出混合结构的主体系统，试图以此来融合经典和非经典的人工智能。

最显然的方式就是在一个主体中包含两个（或多个）子系统：一个是慎思子系统，含有用符号表示的世界模型，并用主流人工智能中提出的方法生成规划和决策；另一个是反应子系统，用来不经过复杂的推理就对环境中出现的事件进行反应。通常，反应子系统的优先级比慎思子系统高，以便它对环境中出现的重要事件提供快速的反应。

最著名的一种混合结构是由 Georgeff 和 Lansky 开发的 PRS (Procedural Reasoning System)。图 14.10 给出 PRS 的结构。和 IRMA 一样，PRS 也是一个“信念—愿望—意图”结构，它也有一个规划库和符号表示的信念、愿望和意图。信念是一些关于外部世界和内部状态的事实 (facts)。这些事实用通常的一阶逻辑表示。愿望用“系统行为”表示（而不是静态的目标状态）。在规划库中包含一些部分完成的规划，叫做知识块 KA (Knowledge Area)。每个知识块和一个激活条件联系在一起。知识块可以被目标驱动的或数据驱动的方式激活；知识块还可以是“反应式”，使主体可以快速对环境的变化作出反应。当前系统中激活的知识块就是它的意图。这些数据由一个系统解释器操纵。系统解释器负责更新信念、激活知识块、执行行动。PRS 系统曾用在航天飞机维护过程的模拟实验中。

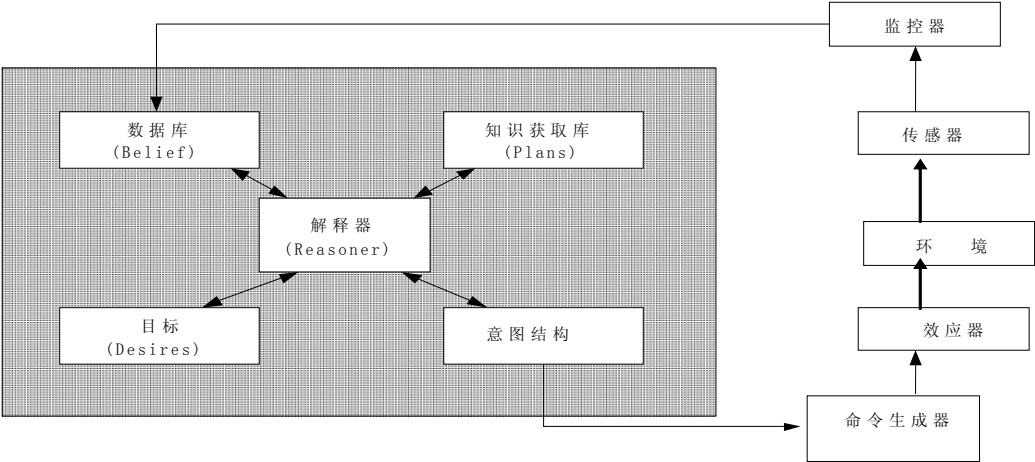


图 14.10 PRS 的结构

另一个著名的混合结构是 Ferguson 的 TouringMachine。该系统中的主体有感知和行动两个模块和外界交互信息，还有三个并行执行的控制层次：反应层 (R)、规划层 (P)、和建模层 (M)。每个层次包含对世界的不同层次的抽象模型，用来实现不同的任务。反应层用来对其它层次无法反应的突发事件作出快速响应。规划层构造规划以实现主体的目标。规划层有两个部件：一个规划器，一个集中注意力的机制。集中注意力的机制主要用来限制提供给规划器的信息量。它将无关紧要的环境信息过滤掉，从而提高规划器的效率。建模层则是主体对其环境中其它实体的模型的符号表示。

在 TouringMachine 中各层次之间是完全独立的，并行进行处理，都可以产生相应的行动。例如，逻辑推理完全由建模层实现，而和其他层次无关。不同层次之间可以互相传递消息，并嵌入在一个控制框架中。控制框架通过使用控制规则处理不同层次提出的有冲突的行动建

议。采用这种结构的系统有 TouringMachine[Ferguson 1991] 和 InteRRaP[Muller 1994]等。在这些系统中，主体的多个控制子系统按层次结构排列，层次越高的子系统处理抽象程度越高的信息。例如，最低层的子系统处理传感器接受到的原始信息，并将之直接映射到效应器的动作上；而最高层的子系统则处理中长期的规划等抽象的问题。这种结构中最关键的问题就是如何将主体的各个子系统适当地嵌入到一个控制结构框架中，以及如何控制各个层次之间的信息交互。

14.4.5 INTERRAP

德国 Fischer、Muller 和 Pischel 研制了一种混合主体结构 INTERRAP，将反应、慎思和协作能力结合起来。Muller 考虑 INTERRAP 的设计原则如下：

- 根据不同程度的抽象和复杂程度，采用三层结构描述主体；
- 不仅控制过程是多层的，知识库也是多层次的；
- 控制过程采用由底向上，即当任务超过本层的能力时，它的上一层就发生控制作用；
- 每层使用下层的操作原语达到它的目标。

INTERRAP 的主体结构主要由世界接口、控制器和知识库构成（图 14.11）。控制器分成三层：行为层（BBL）、本地规划层（LPL）和协作规划层（CPL）。主体知识库是按世界模型、心智模型和社会模型组织的。不同层次对应于主体不同的功能水平。行为层允许主体对一定的外界情况给予反应。主体的世界模型与它的环境对象级知识结合识别事件，触发反应器。本地规划层给予主体长期慎思的能力。协作规划层最后完成一个主体的规划功能，作为协作规划的一部份，这可以进行协作，解决冲突。协作规划层运用世界模型和心智模型的知识，同时使用知识库中社会模型保存的其它主体的目标、技能和承诺。

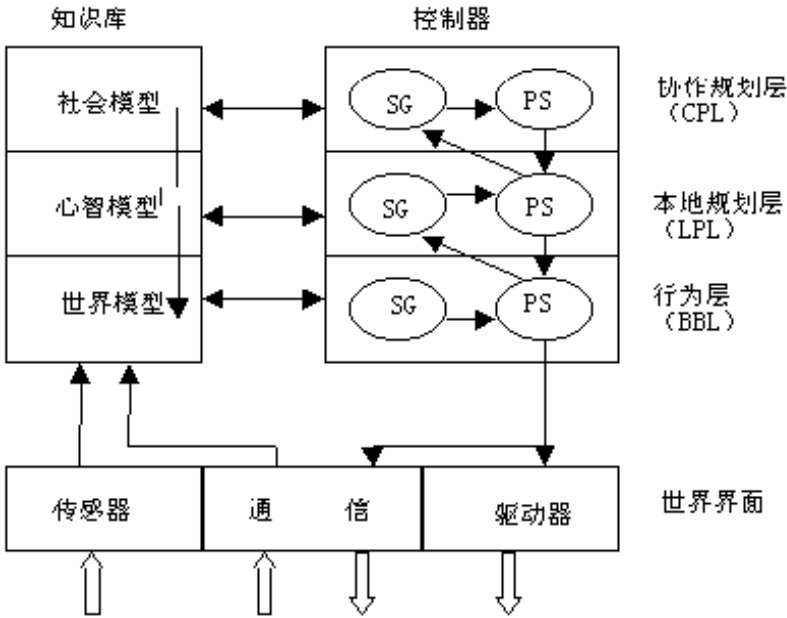


图 14.11 InteRRaP 主体结构

14.4.6 MAPE 主体结构

生活在一个现实或虚拟的世界中的主体，会遇到各种不同的情况。主体除了要保持对紧急情况的及时反应，还要使用一定的策略对中短期的行为作出规划，进而通过对世界和其它主体的建模分析来预测未来的状态，以及通过通信语言实现和其他主体的协作或协商。我们希望这些功能能同时存在，并行地执行，以提供良好的实时性。这些功能基本上是可以独立的，而且每种功能需要采用不同的算法，所以，我们提出一种主体的混合结构，即在一个智能主体中有机地组合了多种相对独立、并行执行的智能形态（如图 14.12 所示）[Shi 1994]。

14.4.6.1 部件构成

MAPE 是一种混合结构的主体，每个主体包含感知、动作、反应、建模、规划、通信、决策等模块。

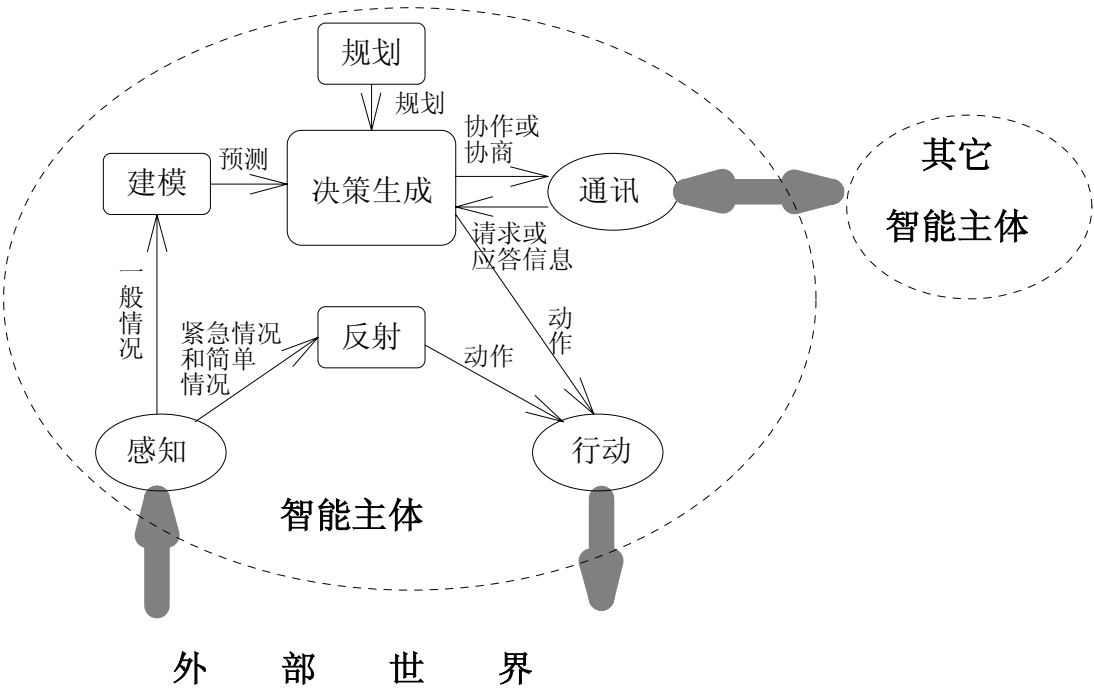


图 14.12 MAPE 主体的混合结构

主体通过感知模块来反映现实世界，并对环境信息作出一定的抽象。根据信息的类型，感知模块将经过抽象的信息送到不同的处理模块。如果感知到的是简单的或紧急的情况，则信息被送到反应模块。反应模块对传入的信息立即作出决定，并将动作命令送到行动模块。行动模块则根据传入的动作命令作出相应的动作，对世界作出影响。从感知行动的信息传递过程构成了“反射弧”。

如果感知到的是复杂的或时间充裕的情况，则信息被送到建模模块进行分析。建模模块中包含主体对世界和其他主体所建立的模型，根据模型和当前感知到的情况，主体可以对短期的情况作出预测，进而提出相应的对策。模型不是一成不变的，主体根据感知的情况、以往的经验修正它对世界和其他主体所建立的模型。

规划模块是一个重要的部分，它根据主体的目标对中短期行动作出规划。规划是一组动作序列，它被提交给决策生成模块。如果不发生意外情况，则主体将按这个动作序列行动。

如果发生意外情况（如出现紧急情况，或预测出的情况与规划时的假设情况不同），则决策模块根据需要对规划模块重新作出规划，或者暂停规划的执行。

通信主要用来处理主体之间的信息交换。通信语言中包括通知、请求、询问、回答、致谢等多种类型的语句。通过使用通信语言交换信息，可以实现多主体的协作或协商。

下面我们详细讨论反应、规划、建模、通信和决策等模块的结构。

1. 反应模块

反应模块的存在就是为了使主体对紧急或简单的情况作出迅速反应，所以在反应模块中基本上不作推理，而是直接由感知的信息映射到某种行动。如果使用知识库的方法，则反应模块由一系列如下的规则构成：

```
RULE-Ri: IF
    感知信息条件子句
    THEN
    行动
```

我们用神经网络来实现反应。神经网络的输入参数是数值化的感知信息和自身状态，输出参数是动作的编码。建立起神经网络后，经过大量样本的训练，使它对一些常见的情况可以作出比较合理的反应，然后投入使用。这样在遇到紧急情况时，主体在从感知到行动的反应过程中实际上已经使用了大量的经验，所以一般会作出更合理的反应。

通过反应产生的动作具有最高的优先级，动作模块将立即执行，而将从决策模块送来的动作中断。如果发生中断，决策模块将决定是重新进行规划，还是继续原来规划好的动作序列。

2. 规划模块

主体的规划模块负责建立中短期的行动计划。主体的规划是一个局部的规划。局部性体现在两个方面。一方面，每个主体根据目标集合、自身的状态、自己对世界和其他主体的模型，以及以往的经验规划自身的行为，而不是由某个主体对全局进行规划并将命令分发给其他主体。另一方面，主体并不需要对它的目标作出完全的规划，而只要生成近期的动作序列就可以了。因为世界是运动的，很多情况无法预料，长期的规划很可能会因为情况的变化而失去意义。如图 14.13 所示，规划模块需要从目标集合、世界的模型、其他主体的模型、经验库，以及自身的状态等数据结构中提取信息，经过局部规划器，产生出近期的动作序列，送交给决策模块。

目标集合包含主体要达到的目标，表示为 G_1, G_2, \dots, G_n 的形式。目标的排列顺序是重要的，它决定了目标的优先级。在其他条件完全相同的情况下，目标优先级的不同可能会产生迥异的规划。世界和其他主体的模型由建模模块提供，它使得主体可以对环境的变化趋势作出预测，并反应到规划中。经验库是一些范例的集合。每个范例由前提条件、规划和结果评价组成。局部规划器总是试图在经验库中找到有和当前情况最为类似的前提条件的范例，然后参考其规划和结果作出新的规划。如果找不到前提条件和当前情况的差异小于某个阈值 S 的范例，则局部规划器只得尝试新的规划，并记录结果。

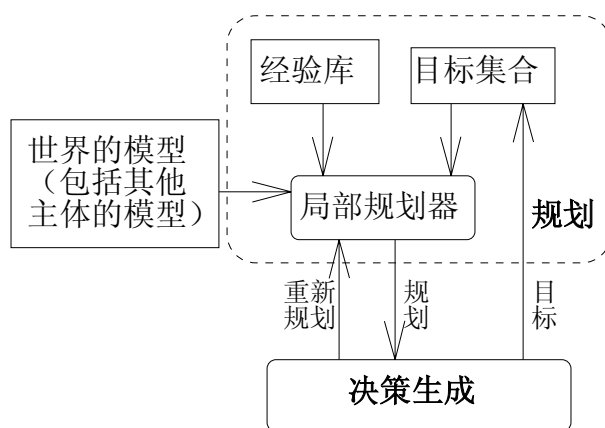


图 14.13 规划模块的内部结构

3. 建模模块

建模模块有两个功能，一是维护和更新主体对世界和其他主体所建立的模型，二是根据当前感知的信息和模型对近期的情况作出预测，并提出行动的建议。图 14.14 表示了建模模块的内部结构。

主体的世界模型只是该主体对世界的认识和反映。这种世界模型开始时并不保证一定正确，也不一定具有全部信息。主体对世界的模型主要包括世界的拓扑知识（例如城市的地图），以及世界的组成部分的物理、化学、生物等方面的性质等信息。对于其它主体的模型，包括主体的位置和性质、信念、目标、能力、关系等信息。正如前面所说，这些信息有可能是错误的或不完全的。主体最初从程序设计人员提供的模型库中得到关于世界的基本模型，然后在生存期间内，通过感知以及和其他主体的通信来修正模型。

模型提供了预测的基础。一方面模型被规划模块用来建立行动计划，另一方面，建模模块使用模型和当前感知信息预测将出现的情况，并将行动的建议提交给决策模块。

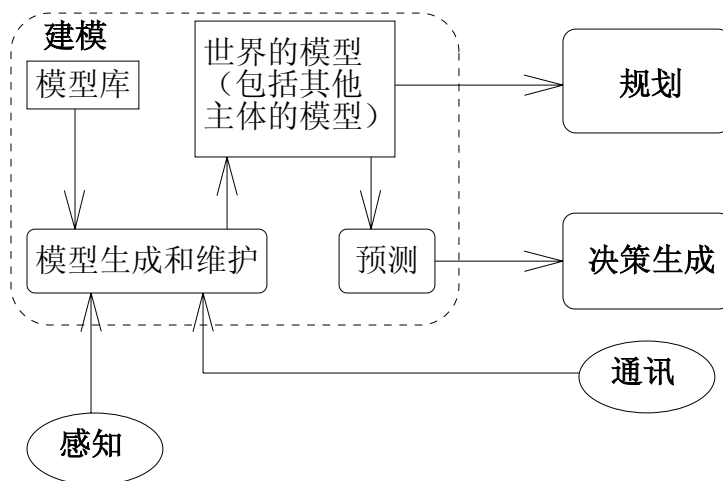


图 14.14 建模模块的内部结构

4. 通信模块

可通信性是主体的基本特征，而且通信语言的完善程度和灵活性直接影响到主体表现出的智能程度。如图 14.15 所示，通信模块包括语言理解、语言生成、物理通信、以及词法库、语法库、语义库等多个部分。

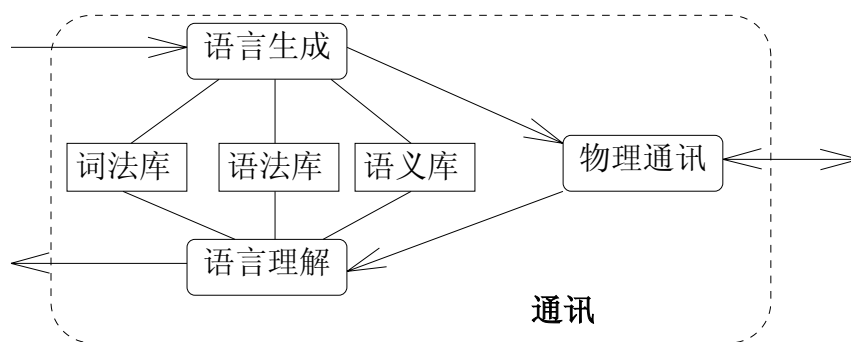


图 14.15 通信模块的内部结构

主体根据词法库、语法库、语义库，对通信语言进行理解，并将一部分抽取的信息送交给决策模块（如其他主体的请求信息）和建模模块（如其他主体的信念信息）。对于一些基本的应答信息，则由通信模块直接作出反应。决策模块生成的和其他主体的协商和交互信息通过语言生成模块变成通信语言。

我们前面所讨论的规划基本上是单主体规划，即规划只针对自己进行。如果要进行多主体的协作，则需要进行多主体规划或者相互规划（Mutual Planning）。关于这方面还有待于进一步研究。

5. 决策模块

决策模块不是对主体的中央控制，而是负责各个模块的协调工作。它的输入有规划模块生成的行动计划、建模模块的预测和行动建议、以及通信模块的请求等。它的主要工作是冲突检查和消解，并决定当前的动作和通信。

世界是随着时间的推移而变化的，所以原定的计划和当前的情况，以及其他主体的请求会发生冲突。这时，决策模块根据一定的规则对冲突进行消解。例如，一个汽车主体正在按计划向预定目的地行驶时，感知到另一汽车占据了原定要经过的一条单行道，建模模块因此建议汽车改道。这时决策模块根据规则，有两种解决方案。一种是先减速行驶，并立即发消息给规划模块让它重新规划；如果时间充裕，则可以决定暂停计划的执行，等另一汽车离开单行道后再继续执行预定计划。

任务的承接也由决策模块完成。任务一般以其他主体的请求的方式进入决策模块。决策模块根据规则决定是否接受任务。如果接受，再决定其优先级，然后修改目标集合，将任务放在合适的位置。如果不接受，则发消息给通信模块让它生成“婉拒”的通信语句。

部件构成

14.4.6.2 主体内核

在 MAPE 中我们提出一种“插件式”地构造主体的方法[289]。我们注意到，虽然在许多方面主体之间是不同的，但是它们确实有一些相同的特性。如通信方式、执行机、心智状态（mental state）的表示等可以是相同的。不同之处仅仅在于作决策的策略、它们能做什么动作、知识的表示等。通过分离这些部分，我们可以定义一个对所有主体都相同的主体内核（Agent Kernel）。在主体内核上定义一个接口，使得与领域相关的决策方法、功能模块等可以方便地联接到主体内核上。图 14.16 对主体内核进行了形象的说明。

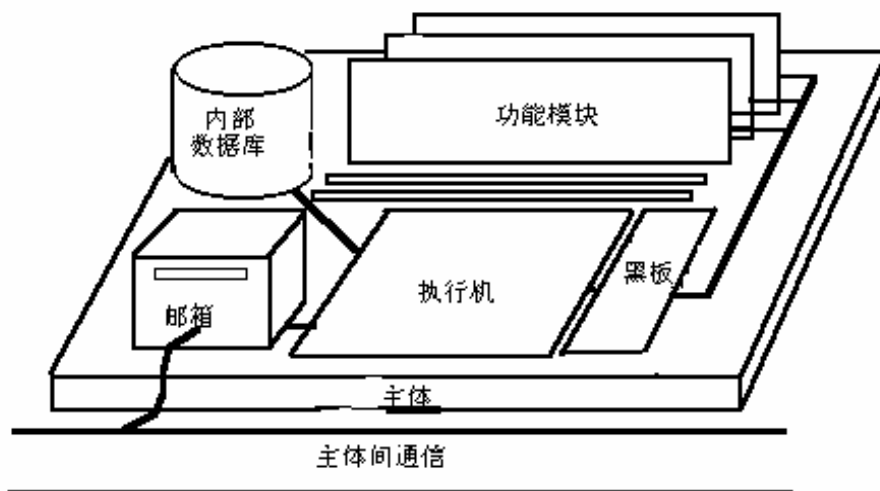


图 14.16 主体内核示意图

每个主体由一个通用的主体内核（Agent Kernel）和许多功能模块（Function Modules）构成。主体内核由内部数据库、邮箱、黑板、执行机等部分组成。其中，内部数据库中包含主体自身的信息、目标集合、世界的模型等信息；邮箱提供主体和环境以及其他主体的通信；黑板提供主体内部各个功能模块之间的通信；执行机则完成消息分派、功能模块的执行控制等。各个功能模块都是相对独立的实体，由执行机启动后即完全并行地执行，并通过黑板协调工作。

利用这样的方法，可以方便地实现我们所提出的复合式结构的主体。感知、行动、反应、建模、规划、通信、决策生成等都以功能模块的形式加入主体中。它们可以使用不同的编程语言、数据结构，只要支持同样的黑板格式就可以了。

下面我们对主体内核的结构进行进一步的说明。

1. 主体的内部数据库

主体的内部数据库包括主体对自身的描述，主体对世界状态的描述，主体对其它主体状态的描述等。

每个主体都有一个描述，形成主体数据库中的数据集合。在抽象的层次上对主体进行描述，主要包括：

主体自身的名字、邮件箱地址等；

主体的能力（即它所控制的功能块）的名字及其描述（关键字、路径、运行参数等等）；

功能模块当前的状态。功能模块可以处于运行态、准备好、完成或阻塞态。

主体的状态。主体的状态可以是检查邮件箱、检查黑板、正在进行规划、正在进行推理或运行功能模块等等。

在一个多主体系统中，主体不是孤立地存在，而是处于一个团体（即“主体社团（Agent Community）”）之中。为了通信并与其他主体协作，每个主体都需要有一个对主体社团的描述。该描述并不是完整的描述，仅需要描述该主体所能感知的信息。这些信息位于主体的内部数据库中，它们不是由系统开发人员预先定义的，而是在主体的生存期间动态建立的。该信息是一些记录的集合。每个记录包括：

主体的名字；

邮箱地址；

与本主体的关系。

通信费用因子。

在实现时，我们采用 C++ 语言定义主体基本类 AGENT，主体的内部数据库就是它的类中

的封装数据。主体类的定义如下：

```
typedef struct capa {          //主体能力结构
    char *name;                //主体能力的名称
    int type;                   //主体能力的类型（可以有 BUILT_IN_FM 、 INVOKED_FM 、
                                EVERLASTING_FM 三种）
    char *path;                //对于 BUILT_IN_FM 为 NULL ；对于 INVOKED_FM 和
                                EVERLASTING_FM，为外部功能模块的运行文件路径
    char *keywords;            //能力的描述关键字
    struct capa *next;
}CAPABILITY;

typedef struct FM_queue {      //功能模块队列结构
    int queue_id;              //功能模块的唯一队列标识符
    int stub_id;               //请求的存根号
    char *request_agent;       //请求的主体名称
    CAPABILITY *func;          //对应的功能模块
    int priority;              //优先级
    int status;                //运行状态，有 READY、RUNNING 、 FINISHED、BLOCKED
                                等值
    pid_t fpid;                //运行中的功能模块的 PID
    char *argument;            //运行参数
    struct FM_queue *next;     //功能模块队列是一个双向队列
    struct FM_queue *prior;
}FM_QUEUE;

typedef struct acquaintance {  //其它主体的信息
    char *agent;               //主体名称
    char *url;                 //地址
    int relationship;          //和本主体的关系因子
    int cost;                  //通信费用因子
    struct addressbook *next;
} ACQUAINTANCE;

class AGENT {                 //主体类
private:
    FM_QUEUE *Queue_head, *Queue_tail; //功能模块队列的队首和队尾指针
    ACQUAINTANCE *Acquaintance;        //其它主体的队列的首指针
    int Queue_count;                    //当前功能模块队列的长度
    BBBUFFER *Blackboard;              //黑板指针
    int Blackboard_key;                 //黑板关键字
    int Status;                         //主体的状态
public:
    char *Name;                        //主体的名字
    char *Url;                         //主体的地址
```

```

char *Facilitator; //通信服务员的地址
CAPABILITY *Capabilities; //主体的能力队列首指针

AGENT(char *AgentDescriptionFile); //主体构造函数
~AGENT(); //析构函数
void CheckMailbox(); //检查邮箱函数
void CheckBB(); //检查黑板函数
void Action(); //复合式主体的内置功能模块：行动
void Planning(); //复合式主体的内置功能模块：规划
void Modeling(); //复合式主体的内置功能模块：建模
void Decision_Making(); //复合式主体的内置功能模块：决策生成
int SendMessage(char *ToWhom, char *Performative, char *Content); //发送消息
char *LookupAcquaintance(char *agentname); //在本地查找主体地址
void UpdateAcquaintance(char *agentname, char *url, int relationship, int cost);
//更新或添加其它主体信息
};

```

2. 主体执行机

该部分的功能类似于操作系统中的进程管理，但是其管理的目标是功能模块。每个主体执行相同的控制循环，步骤如下：

初始化。初始化工作包括：

读入主体定义文件；

启动主体间通信；

初始化黑板；

初始化各种队列；

将所有的 EVERLASTING_FM 以 READY 态加入功能模块队列。

检测邮件箱。如果不空，取出第一条消息。

根据消息的类型执行相应的动作。消息遵循 SACL 语言所规定的语法和语义，有 INFORM、REQUEST、INQUIRE、RESULT、ACKNOWLEDGEMENT 等等类型。例如：

如果消息是 INFORM 类型，则将其内容放在黑板上以便所有功能模块可以读取；

如果消息是 REQUEST 类型，则表示对方请求本主体的某个功能模块执行某种操作。这时如果该功能模块的状态是 RUNING，则将该消息通过黑板传递给它请求的功能模块；如果该功能模块还没有启动，则将该功能模块以 READY 态插入功能模块队列的尾部；如果该功能模块是被 BLOCKED，则将它激活；

如果消息是 INQUIRE 类型，则表示它要询问本主体的能力或状态，这时应根据情况给出回复；

如果消息是 RESULT 类型，则表示它是本主体的内核或某个功能模块询问或请求的结果，应该根据情况保存或传递给相应的功能模块。如果该功能模块处于 BLOCKED 态等待这个结果，则还要将它激活；

如果消息是 SHUTDOWN 类型，则表示通信服务器要求本主体中止运行。这通常是操作人员的意愿，一般应遵守，转第 7 步。

检测黑板。如果有消息需要发出，把该消息发出并为其赋上适当的消息类型。如果是 END_AGENT 类型，则表示某个功能模块请求本主体停止运行。如果接受这个请求，则转第 7 步。

检测功能模块队列，选择适当的功能模块执行。功能模块将在后台一直运行，直到它终

止或被自身阻塞。

转到第 2 步。

停止运行。这包括：

如果还有功能模块在运行，则首先中止这些进程；

向通信服务器发出通知并中止通信；

保存信息、释放内存等。

3. 功能模块的状态转换

一个主体可以有多个功能模块。这些功能模块都是预先编译好的可执行代码。它们通过黑板和主体内核交互信息。在 MAPE 中，我们还提供了一套用来操纵黑板的编程接口函数库。在功能模块中使用这些函数就可以完成和主体内核的通信。

功能模块是一些进程，它们的状态由自身和主体内核一起控制。图 14.17 是功能模块的状态转换图。虽然其中的状态的名称和操作系统中使用的进程状态名称相同，但是它们的意义并不一定相同。

“静止态 (Static)”就是该功能模块没有运行而只是停留在磁盘上时的状态。这个状态在 FM_QUEUE 中并不会出现，所以用虚线表示。

“就绪态 (Ready)”就是主体内核将该功能模块作为一个结点加入功能模块运行队列的尾部时的状态。这时该功能模块仍然停留在磁盘上。

主体内核的执行机的第 5 步是“选择适当的功能模块执行”。同时可能有很多处于“就绪态”的功能模块，主体内核将根据一定的规则（如优先级、请求主体的关系因子等）选择一个功能模块执行。所以一个处于“就绪态”的功能模块并不一定立即被执行。直到主体内核选择了它，它才进入“运行态 (Running)”，即真正作为一个进程出现在系统中。功能模块运行结束后就仍然回到“静止态”。

如果功能模块需要等候一个请求的结果，它可以将自己阻塞，进入“阻塞态”。阻塞态的功能模块将不能进行任何计算，一直到它被激活。

被自己阻塞的功能模块必须由主体内核激活。在接收到对它的请求或者给它的结果时，主体内核将激活它。被激活的功能模块直接恢复到运行态。

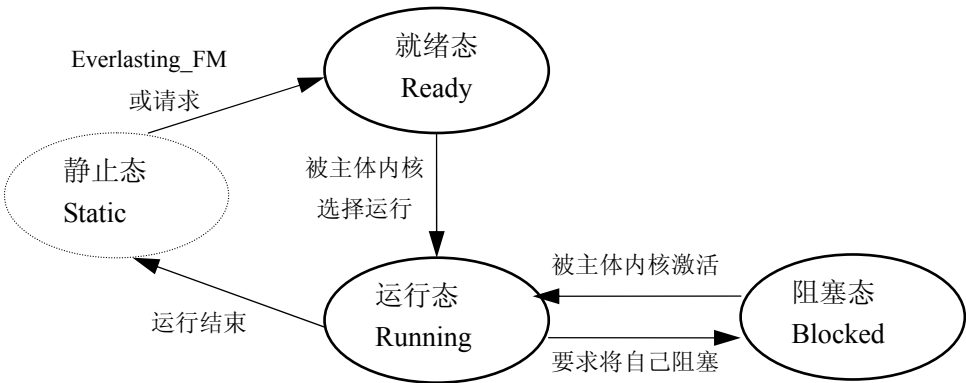


图 14.17 功能模块的状态转换图

4. 主体与功能模块之间的接口

主体内核与功能块之间的通信是通过黑板完成的。我们为内核与功能模块的通信提供了一套标准的编程接口，使得功能模块可以很方便地实现和内核以及其它主体的通信。这样做的重要意义在于，原来编制的任何程序只要做少量修改就可以作为任何一个主体的功能模块。这使得用户不需要每次为主体开发新的程序，从而很好地实现了软件的复用和移植。这正是

“插件式”构造主体的方法的核心精神。

更进一步，如果积累了大量的功能模块，那么可以形成一种比“类库”更为高级的面向问题的“功能模块库”。这些模块可以随意组合到主体内核上形成具有某种功能的主体。

A. 黑板

一个主体可以有多个功能模块。这些功能模块都是预先编译好的可执行文件。主体内核也是一个可执行文件。它们运行在不同的地址空间之中。它们之间交换信息必须通过共享内存或者共享文件的方式。由于共享内存操作比共享文件操作要快得多，而且更易于进行并发控制，所以我们采用共享内存的方式来传递主体内核与功能模块，以及两个功能模块之间的消息。这块共享内存就被命名为“黑板”。

黑板结构和处理函数由“bbapi.h”定义，所以主体内核和每个功能模块都应该包含这个头文件，才能进行有关黑板的操作。

黑板是一个很大的共享内存区域，其中划分为一些同样大小的区域。这些区域用来存放消息，称为“黑板条目”(Blackboard Item，简称 BB_ITEM)。黑板条目的定义如下：

```
typedef struct {
    char flag;
    int type;
    int stub_id;
    int queue_id;
    char dest[AGENTNAMESIZE];
    char content[CONTENTSIZE];
} BB_ITEM;
```

其中：

flag 表示该条目当前是否正被使用，其值可以为 USING 和 NOT_USING。

type 表示该条目中的信息的类型，通常由表示信息源和目的和表示信息的功能的两部分组成。预先定义的值有：

表示信息源和目的的常量：

FROM_KERNEL_TO_FUNC 从内核到功能模块

FROM_FUNC_TO_KERNEL 从功能模块到内核

FROM_FUNC_TO_FUNC 从功能模块到（另一个）功能模块

表示信息的功能的常量：

ENDFUNC 结束功能模块

TELL_PID 将功能模块的 PID 通知内核

BLOCKFUNC 阻塞功能模块

ENDAGENT 结束功能模块，同时结束整个主体

RESULTTYPE CONTENT 中是结果信息

REQUESTTYPE CONTENT 中是请求信息

PINGAGENT 查看某主体是否存在

FINDAGENT 按功能模块的关键字查找主体

OTHER 非保留类型，由功能模块解释

例如，功能模块要向内核通知 PID，使用的值是

FROM_FUNC_TO_KERNEL | TELL_PID

stub_id 是一个整型域，它在不同的消息类型之下有不同的意义：在 RESULTTYPE 和

REQUESTTYPE 中，它表示结果或请求的存根号；在 TELL_PID 中，它表示功能模块的 PID。

queue_id 用来指示功能模块在内核的执行队列中的位置。

dest 用来存放一个主体的名称。

content 用来存放消息的内容。

黑板的结构定义如下：

```
typedef struct {  
    int stub_count;  
    char semaphore;  
    BB_ITEM bb[BBNUMBER];  
} BBBUFFER;
```

其中：

stub_count 用来记录当前的存根号，以便生成新的唯一的存根号。

semaphore 是一个信号量，通过它可以封锁对黑板的读写，以避免因并发操作而引起的错误。它的值可以为 USING 和 NOT_USING。

bb 是 BBNUMBER 个黑板条目的数组。

B. 功能模块

功能模块 (Function Module, 简称 FM) 是一些预先编译好的可执行文件，它们和主体内核一起构成了一个主体。功能模块通过黑板和主体内核或其他功能模块交换信息。功能模块可以有两种类型：

(1) INVOKED_FM 这种功能模块在主体启动时并不同时启动，而只是登记一下它的存在。如果有别的主体或者同一主体内的别的功能模块请求它的功能，则主体内核负责启动它。一般说来 (但是并不一定这样)，这种功能模块在执行完任务后就结束运行，直到有另一个请求时再次由主体内核启动它。

显然，这种功能模块的好处在于它在没有任务时不占用系统资源。它适合于进行不需要交互的单项计算或者操作，而不太适合于交互式操作或有图形界面的功能模块。

(2) EVERLASTING_FM 这种功能模块在主体启动时同时启动。一般说来 (但是并不一定这样)，它循环检查黑板上是否有给它的消息，如果有则进行相应的处理。通常这种功能模块也会和主体一起结束运行。如果是这样，则在结束之前，它可以调用 “End_agent” 函数通知主体内核结束运行。如果主体内核已经要结束运行时，还有功能模块没有结束，则主体内核会杀死所有功能模块。

和 INVOKED_FM 相反，这种功能模块的好处在于它适合于交互式操作或有图形界面的功能模块；而缺点正在于它在没有任务时仍然占用系统资源。

主体内核在启动功能模块时将赋予它一个唯一的 “queue_id”。这个 queue_id 相当于功能模块的 “身份证”，功能模块在黑板上传递消息时必须使用它来表明自己的身份。另外，功能模块要利用黑板通信，必须将黑板共享内存附着在它的地址空间中。要附着共享内存，必须知道它的 “关键字”。所以主体内核在启动功能模块时要通过命令行传递一些参数给功能模块，而以后的所有通信都通过黑板实现。

根据两种功能模块的不同用途，主体内核在启动它们时使用不同的命令行参数：

对于 INVOKED_FM，命令行参数是：

```
argv[0] -- 字符串，功能模块的运行文件名  
argv[1] -- 整型，功能模块的 queue_id  
argv[2] -- 整型，黑板关键字
```

argv[3] -- 整型, 请求的存根号 (stub_id)
argv[4] -- 字符串, 请求主体的名字
argv[5] -- 请求参数 (类型不定, 由功能模块自己解释)

对于 EVERLASTING_FM, 命令行参数是:

argv[0] -- 字符串, 功能模块的运行文件名
argv[1] -- 整型, 功能模块的 queue_id
argv[2] -- 整型, 黑板关键字

C. 通过黑板进行信息传递

MAPE 系统中为用户提供了很多用于通过黑板进行信息传递的函数。创建主体时编程规则和注意事项如下:

功能模块启动后应该调用 Init_FM, 该函数将进行附着黑板, 向主体内核报告 PID 等操作;

功能模块结束时应该调用 Shutdown_FM, 该函数将进行释放黑板, 向主体内核报告功能模块结束的消息等操作;

如果功能模块结束时同时希望主体结束运行, 应该调用 End_agent, 该函数将释放黑板、向主体内核报告功能模块结束的消息、同时请求主体内核结束运行;

作为客户方, 要请求另一个主体的某个功能, 应该使用 PutRequestToBB 函数。该函数返回一个请求存根。然后用 GetResultFromBB 函数取回该请求的结果;

作为服务方, 可以用 GetMessageFromBB 函数取回一个对它的请求消息 (如果是 INVOKED_FM, 则在启动时的命令行参数中可以获得第一次请求的消息)。然后用 PutResultToBB 送出结果消息;

任何一个主体, 主体中的任何一个功能模块都可以为客户方, 也可以为服务方, 或者同时既是客户方又是服务方。

5. 主体定义文件

主体定义文件是一个纯文本文件, 其缺省的名称应该是 “<主体名>.cfg”。主体内核启动时如果没有运行参数, 则它将寻找缺省的主体定义文件。如果有参数, 则以参数作为主体定义文件的名称。

主体定义文件可以由用户用手工填写, 也可以用主体构造工具自动生成。主体定义文件定义主体的各种重要参数, 包括:

AGENTNAME = <主体名称>
AGENTURL = <本主体的 URL>
FACILITATOR = <通信服务器的 URL>
CAPABILITY = <功能名称>, <类型>, <运行文件路径>, <关键字>[, <关键字>, ...]
BLACKBOARDKEY = <黑板关键字 (整型值)>

14.5 主体通信语言 ACL

智能物理主体基金 FIPA (Foundation for Intelligent Physical Agents) 是一个在瑞士日内瓦注册的非赢利协会 [FIPA 1996]。它的目的是为了促进基于主体的应用、服务和设备的成功实现。FIPA 通过制定能及时获得的国际承认的规范来达到这一目标, 这种规范可以最大限度地增大基于主体的应用、服务和设备的互操作性。截止 1999 年 4 月, FIPA 已有 50 多个合作成员, 他们来自全世界 12 个国家。

FIPA97 规范定义了一种语言和支持工具，如协议，用于智能软件主体相互通信。软件主体技术对这类主体采用了一种高层的观点，它的许多思想都来源于采用其它方式的社会交互作用，例如人与人之间的通信。本规范并非试图定义通常与分布式软件系统之间通信相关的低层和中间层的服务，例如网络协议，传输服务等。事实上，用于物理传输组成交互主体通信动作的比特序列的这类服务都是假设存在的。

对于软件主体没有单一的、通用的定义，但是主体行为的一些特性是广泛接受的。FIPA 定义的通信语言用于支持和促进这些行为。这些特性包括但不限于这些：

目标驱动行为

动作过程的自主决定

通过协商和委托进行交互

心智状态模型，例如**信念、意图、愿望、规划和承诺**

对于环境和需求的适应性

14.5.1 主体间通信概述

抽象的主体特性，即没有预先规定任何具体的主体执行模式和认知结构，可以用主体的心智态度描述。在 FIPA 规范中，主体的心智态度描述如下：

(1) **信念** 表示一组主体接受为真的命题，接受为假的命题表示为相信这个命题的否定。

(2) **不确定性** 表示一组主体不能肯定为真或假的命题，但是更倾向于真，而更倾向于假的命题表示为命题否定的不确定。值得注意的是，不确定性并没有妨碍命题采用具有某种程度支持命题的特别不确定信息形式，例如，概率论。更准确地说，不确定性为主体提供了一种利用不同表达方案，讨论不确定信息的最小承诺机制。

(3) **意图** 表示一种选择，或者主体愿望为真和目前不认为是真的一个或一组社会特性。接受了这种意图的主体将形成一个行动计划，而这一行动计划将导致它的选择所指示的社会状态的产生。

对于一个给定的命题 P，相信 P，相信非 P，P 的不确定以及非 P 的不确定等态度是相互排斥的。

此外，主体理解并且能够执行一定的行动。在分布式系统中，一个主体仅能通过影响其它主体的行为来实现自己的意图。

对其它主体行为的影响是由一种行为的特殊类，称为通信动作的行为实现的。通信动作是由一个主体向另一个主体实施的。执行一个通信动作的机制，简要地说就是发送编码动作消息的机制。因此，通信动作初始者和接受者的角色通常分别表示为消息的发送者和接收者。

FIPA 定义的消息由一组精心定义的核心集合产生出来，它表示一组通信动作，而这组通信动作试图平衡定义的一般性、表达能力和简单性，以及对于主体开发者的易懂性之间的关系。消息类型定义了被执行的通信动作。结合适当的领域知识，通信语言可以使接受者确定消息内容的含义。

关于发送者的心智态度和从发送者的观点对接收者心智态度产生的期望结果在通信动作中是用前提条件这个词来表达的。但是，因为发送者和接收者都是独立的，所以期望的结果是没有任何保证的。

两个主体通过交换消息彼此进行通信，它们必须有一个消息传递公共会合点。消息传输服务属性的审议事项是由 FIPA 技术委员会 1——主体管理进行的。

主体技术对于复杂系统行为和交互操作的贡献尤其表现在高层交互上，FIPA 提出的 ACL 是以这一观点为基础。例如，本文描述的通信动作用于通知已相信的事实、请求复杂动作、

协商协议等。这里提到的交互机制不能与低层的网络协议例如 TCP/IP, OSI 七层协议模型等相竞争, 也不应与它们相比较。它们也不是 CORBA、Java RMI 或者 Unix RPC 机制的替代品。然而它们的作用在某些方面与上面的例子相重叠, 至少 ACL 消息通常用这种机制来传递。

考虑了 FIPA 通用开放主体系统目标以后 ACL 的作用将会更加清晰。其它机制如著名的 CORBA 也有此目标。但是, 实现时在对象表示的界面上强加了一些限制。历史经验告诉我们: 主体和主体系统可以采用多样的界面机制典型地实现; 已有的主体例子包括使用 TCP/IP 套节字, HTTP, SMTP 和 GSM 短消息的主体。ACL 试图在消息传递服务上将要求减到最小以尊重这种多样性。特别地, 最小消息传输机制被定义为: 通过简单比特流传递的一种文本形式, 这也是被广泛采用的 KQML 主体通信语言所采用的方法。这种广泛的方式用在非常高性能系统上, 这些系统的消息生产率非常高。FIPA 以后将定义可替代的传输机制方案, 包括其它传输语法, 它们将满足非常高性能的系统要求。

目前, ACL 在主体传输服务上定义了如下的一组最小要求:

消息服务能够传递一个与比特序列一样的消息到目的地, 通过它的界面消息服务能发现它是否能可靠地处理高阶位被设置的 8 位比特流。

正常情况下消息服务是可靠的 (包装好的消息能达到目的地)、准确的 (接收的消息在形式上和发送的一样)、有序的 (从主体 a 发送到主体 b 的消息到达 b 时和它从 a 发送的顺序一样)。除非特别说明, 一个主体将被认为具备这三个特性。

如果消息传递服务不能保证以上一个或所有的特性, 它将通过消息传递服务的界面以某种方式表示出来。

主体将能选择是否暂停和等待消息结果或者在等待消息回复时继续其它无关任务。这种行为的有效性是执行细节, 但是这种行为是否支持必须明确。

传递消息动作的参数, 例如, 如果没有回复时间超出, 不是在消息这一级说明, 但是它是消息传递服务界面的一部分。

消息传递服务将发现和报告出错情况并返回给发送主体, 例如消息出错形式, 不可传递, 主体找不到等。依据错误情况, 它将返回一个消息发送界面的返回值, 或者通过一个相关出错信息的传递来返回。

一个主体将有一个名字使得消息传递服务能将消息传到正确的目的地。消息传递服务能够决定正确的传输机制 (TCP/IP, SMTP, HTTP 等), 允许主体位置的变化。

14.5.2 FIPA ACL 消息

FIPA 定义单独的消息类型, 特别是消息的格式和消息类型的含义。消息类型对本规范定义的语法规则是一个参考, 这些类型对于整个消息及动作和消息内容都赋予了一个意义。

例如, 如果 i 通知 j “Bonn 在德国”。那么从 i 到 j 的消息内容是 “Bonn 在德国”, 而动作则是通知这一行为。“Bonn 在德国” 有一定的意义, 而且在对 “Bonn” 和 “德国” 这两个符号进行任何合理解释的情况下它将是真的, 但是消息的意义包含了对主体 i 和 j 的作用。决定这些作用的本质上是 i 和 j 的私人事物, 但是由于这个有意义的通信即将发生, 所以对这些作用的合理期望将会得到满足。

显然, 消息的内容涉及的领域知识将不受限制。ACL 没有强制任何表示消息内容的形式化方式。主体本身必须能够正确地解释任何一种已知的消息内容。ACL 规范的以后版本将讨论本版本没有讨论本体论共享问题。本规范所宣布的是规范独立于内容的规则含意。一组标准通信动作和它们的含意都详细地定义了。

值得注意的是在动作的能力和规范之间有一种平衡。抽象地说, 一组大量的能传达细微

意思差别的非常详细的动作类型和一组少量的更通用的动作类型效果可能一样，但这组动作类型将对主体有不同的表示和执行限制。这组动作类型的目标可表示为：①全面覆盖大范围的通信情况；②不要使主体的设计负担过于繁重；③在为主体提供通信动作使用选择服务时，减少冗余和模糊。简而言之，定义 ACL 语言的目标是：完整、简单和简明。

ACL 中消息的基本观点是消息表示为一个通信动作。为了优雅和一致，在对话时处理通信动作应和处理其它动作相一致；一个已知通信动作是一个主体能完成的动作中的一个。术语消息在本文中根据上下文表示两个不同的含意：消息可以是通信动作的同义词，或者是指传递主体言论到目的地的消息传输服务使用的计算结构。

ACL 规范提出的通信语言基于一种精确的形式语义学，它给出了通信动作的一种清晰含意。实际上，这种形式化基础还补充了用于易于高效交互通信实际执行的实用扩充部分。在此基础上，以下定义的消息参数用描述形式加以定义。同样，主体可能用到的规范，如消息交换协议，都只给出了描述形式的操作语义。

1. 主体的要求

这里介绍一组所有主体都能使用的预定义的消息类型和协议。但是，对于所有主体并不需执行所有消息。以下列出了 FIPA ACL 兼容主体的最小要求：

- (1) 主体在收到不认识或者不能处理的消息内容时能发出 not-understood。主体必需能接收和适当处理来至其它主体的 not-understood 消息。
- (2) 主体可以选择执行所有的预定义的消息类型和协议或其中任意一子集。这些消息的执行必须与提供的动作语义定义相一致。
- (3) 使用本文中名字被定义的通信动作的主体必须在执行时与它们的定义相符。
- (4) 主体可以使用本文没有定义的其它名字通信动作，有责任保证接收主体理解动作的含意。但主体不应定义与已定义好的标准动作含意相匹配的新动作。
- (5) 主体必须能正确地将语法形式良好的消息生成与发送的消息相一致的传输形式。同样，主体也必须能将传输语法形式良好的字符序列翻译成相关消息。

2. 消息结构

图 14.18 给出了 ACL 消息的主要组成元素。

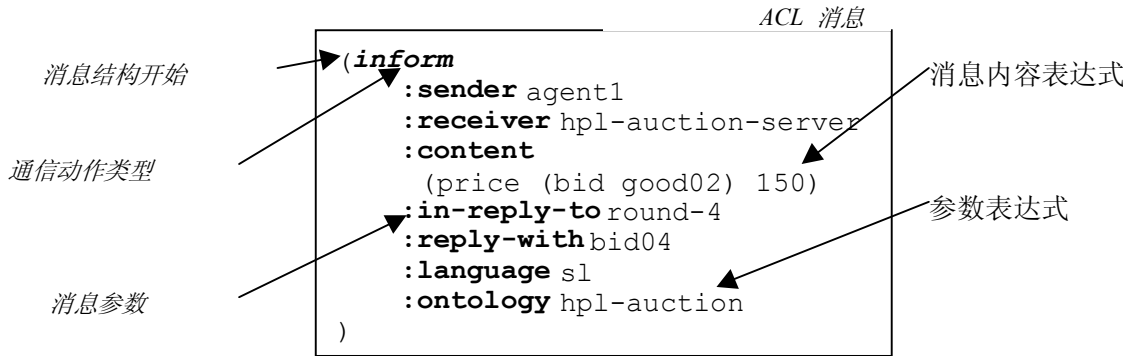


图 14.18 消息结构

在传输形式上，消息表示为 s-表示形式。消息的第一个元素是确定通信的通信动作和定义消息的主要含意。接下来是一列由冒号开头参数关键字引导的消息参数，冒号与参数关键字之间没有空格。其中一个参数包含用某种形式编码的消息内容，其它参数用于帮助消息传输服务正确传递消息（如 sender ,receiver），或帮助接收者解释消息含意（如 language, ontology），或者帮助接收者能更合作地回复（如 reply-with, reply-by）。

传递形式是比特流的序列化，由消息传输服务传递。接收主体负责对比特流解码，解析消息元素并正确处理它。

消息的通信动作类型与 KQML 中的 performative 相对应。

3. 消息参数

如前所述，消息包括一组一个或多个参数。这些参数在消息中可按任何顺序列出。所有参数中只有一个:receiver 参数是必须要的，这样消息传输服务才能正确传递消息。显然没有接收者的消息是没用的消息。但是，有效通信确切地需要哪些其它参数还是根据情况不同而不同。

预定义的消息参数全集如下所示：

消息参数	含意
:sender	消息发送者
:receiver	消息接收者
:content	消息内容
:reply-with	回答原来消息
:in-reply-to	消息回答的原来动作
:envelope	消息服务内容
:language	动作内容的编码方案
:ontology	内容表达式符号的含意
:reply-by	消息发送时间和日期
:protocol	使用协议
:originator	代理主体请求消息
:reply-to	表示向哪个主体回答
:protocol “(” <protocol name>+”)”)	使用协议嵌套
:conversation-id “(” <id>+”)”)	进行的通信动作序列, 表中是子协议
:conversation-id	进行的通信动作序列

4. 消息内容

消息内容表示通信动作申请什么。如果通信动作认为是一个句子，内容是句子的语法元素。内容一般用一种语言编码，这种语言在 language 参数中说明。对内容语言的要求是它应具备以下特性：

(1) 一种内容语言必须能表示命题，对象和动作。虽然任何已有的语言可以更多其它特性，但在这里并不需要其它特性。语言的内容必须表示动作的数据类型：通知的命题，请求的动作等。

命题：表示一种语言中的某些句子是真或假。对象：代表一种讨论领域中可确认东西（可能抽象或者具体）的结构。对象并不一定指面向对象语言如 C++ 和 Java 中出现的专业编程结构。动作：一种主体将解释为由某些主体完成行为的结构。

最一般的情况是主体共同协商一种内容语言。但是，主体必须克服为了商定内容语言而第一步开始对话的困难。这样必须有一个双方都事先知道共同参考点。所以为了注册一个目录服务和执行其它关键主体管理功能，本规范包括以下内容语言定义：

(1) FIPA 规范的主体管理内容语言是 s-表达式概念用于表示属于主体生命周期管理的命题，对象和动作。表达式的术语在本规范的第一部分已定义。

(2) 主体需要通过使用主体管理内容语言和本体论的消息执行标准主体管理能力。语言和本体论在 fipa-agent-management 预留字段相应参数中表示。

5. 消息内容的表示

一个消息的内容是指通信动作中的领域表达式。它是在消息中以:content 参数值进行编码。FIPA 规范没有强制使用一种特定有标准化基础的内容编码语言。

为了提供简单语言编码服务，ACL 语法包括的 s-表达式形式允许任意长度和复杂度的 s-表达式结构。因此一种被定义为通用 s-表达式语法的子语法的语言可不需修改地接受为合法的 ACL 消息。然而，主体通常需要在消息体中嵌入一个用一种符号编码的表达式，而不是用于消息本身的简单 s-表达式形式。ACL 语法提供两种机制，它们都要避免需要 ACL 解释器解释任何语言的表达式问题：

(1) 用双引号包括表达式，这样使它成为用 ACL 语法的字符串，用反斜线符号区别双引号体内的双引号。注意内容表达式中的反斜线字符也需要区别开。例：

```
(inform :content "owner (agent1 ,\" Ian\" )"
  :language Prolog
...)
```

(2) 在表达式前加适当长的编码字符串，这样确保表达式被处理为与它结构无关的词汇符号。例：

```
(inform :content #22 "owner (agent1, "Ian")"
  :language Prolog
...)
```

因此，ACL 解释器将生成一个表示全部嵌入语言式的词汇符号，一个字符串。一旦消息被解释，表示内容表达式的符号通过 :language 参数根据编码方案能被解释。

6. 通信动作的目录

表 14.1 通信动作的目录

Communicative act	Information passing	Requesting information	Negotiation	Action performing	Error handling
accept-proposal			✓		
Agree				✓	
Cancel				✓	
Cfp			✓		
Confirm	✓				
Disconfirm	✓				
Failure					✓
Inform	✓				
Inform-if (macro act)	✓				
Inform-ref (macro act)	✓				
Not-understood					✓
Propagate				✓	
Propose			✓		
Proxy				✓	
Query-if		✓			
Query-ref		✓			
Refuse				✓	

Communicative act	Information passing	Requesting information	Negotiation	Action performing	Error handling
Reject-proposal			✓		
Request				✓	
Request-when				✓	
Request-whenever				✓	
Subscribe		✓			

表 14.1 给出了通信动作的目录。

14.5.3 交互协议

目前主体之间的会话常常形成典型模式。在这种情况下某些消息序列是可预知的，而且在会话的任意一点其它消息也可预知。这些消息交换的典型模式称为协议。主体系统的设计者能选择使主体充分地理解消息的含意、目标、信念和主体具备的其它心智状态，主体规划过程可使这种协议本能地从主体的选择产生。这样使主体执行时的能力和复杂性会有一个沉重负担，虽然在普遍的主体社会中这不是一个通用的选择。一种很实际的替代观点是预先规范这些协议，这样一个简单主体执行这种协议，就能与其它主体进行有意义的会话。

应注意，主体可能与不同的主体同时地进行多方对话。Conversation 项用于指示这种对话的任意特殊实例。这样主体可以同时与多个主体采用不同协议进行多方对话。此部分所指在某种已知协议控制下接收消息的说法只是指特殊的会话。

1. 协议操作约定

两个主体在使用某个协议之前应先协商使用哪种协议。因此采用以下规定：在消息中 protocol 参数写入协议名称，相当于 inform 主体 i 试图采用此协议。一旦协议结束，可能协议达到最后阶段或者协议名称就从消息 protocol 参数中去掉。

如果主体不能或不愿意支持接收到消息中的协议，它可以返回拒绝消息：

例：

```
(request :sender i
      :receiver j
      :content some-act
      :protocol fipa-request
)
```

2. 交互协议的规范

在 70 年代，结构化编程是主要范型。为了方便和规范从意向开始到最后产品维护的全部开发过程，软件工程技术发展起来。八十年代是具有封装和继承特性数据的面向对象的十年。现在统一建模语言（UML）是面向对象工程的现行标准。

面向主体软件和软件开发是今天的范型。与对象相比较，主体由自身产生的原因执行动作而具有主动性。主体的主动性基于它们内部的状态，这些状态包括目标和隐含的已定义任务执行的条件。对象需要从外部控制执行方法，而主体知道条件和行动的意图，因而能满足对它们的需求。而且，主体不只是一个单独行动而是和其它主体一起，其中一个原因是一个主体所有的资源不足时，可以得到其它主体资源的支持。还有，主体可能不具备完成某一复杂任务的所有能力，可以与其它主体合作，协同工作，以完成规定的任务。尽管主体代表各

自用户和分别行动，但在多主体系统中每个成员相互协作，形成一个类似社会集体。

现在研究和开发的主要焦点是建立各种多主体系统的开发平台。但是建立基于主体软件的模型还缺乏必要的规范标准。为了将基于主体编程应用于生产的项目中，一个支持整个软件工程过程的规范技术是必要的。这个过程开始于系统需求描述的使用方案，描述概念的设计规范和系统设计，最后结束于软件产品完成的执行规范。所以，FIPA 提出一份关于交互协议规范作为标准 UML 补充的建议。下一步在 FIPA 2000 报告中将提供一个技术规范，它支持整个面向主体软件工程过程，它将成为 FIPA 规范中极有参考价值的一部分。

在定义交互协议和以此目的形成一份技术规范之前，对于交互协议必须建立一个共识。一个交互协议，缩写为 IP，定义包括两部分：

(1) 通信模式定义，包括

定义一种允许传递消息和应答的策略，其中
参与者具有特别角色，
消息内容有一些语义限制，
对于参与者有固定的规则。

(2) 在 CA（通信动作）语义学中嵌入这种模式，即模式内消息内容与 CA（通信动作）语义学相一致。

以下两概念必须区别：

(1) 基本交互协议 IP 和它们的实例；

(2) 领域相关交互协议 IP。

在 FIPA 框架内规范了基本交互协议 IP，应用程序的开发人员可以在他们自己的系统规范中使用它们。在交互协议 IP 规范中满足任何其它规范技术要求的技术是必要的：

(1) 形式化的、直觉的语义学；

(2) 允许精确定义；

(3) 软件工程过程中可用；

(4) 对于不熟悉面向主体编程的人员可以使用图形符号进行讨论。

交互协议的规范技术必须支持：

(1) 主体的角色；

(2) 消息发送可以具体定义：

消息的基数和多重接收处理；

消息发送的约束，特别是时间约束；

消息的递归和重复。

(3) 子协议的定义，即通信模式的体系结构；

(4) 子协议的递归和循环；

(5) CA 的意图的考虑，即对策中的固定规则；

(6) 消息内容的定义和限制；

(7) 定义并行性和选择；

(8) 描述协议中断，以便执行其它主体的另一种协议。

14.5.4 ACL 语义学的形式化基础

1. 形式化模型

这里给出消息语言语义学基础的通信动作模型。这一模型只是为了给出通信动作和协议的已列出含意的基础。它不是一个建议的体系结构或是一个主体设计的结构模型。也不是一

个单独运作和只与人或与其它软件界面交互的主体特例。主体必须相互通信来实现它们担负的任务。

图 14.19 给出一个基本事例：

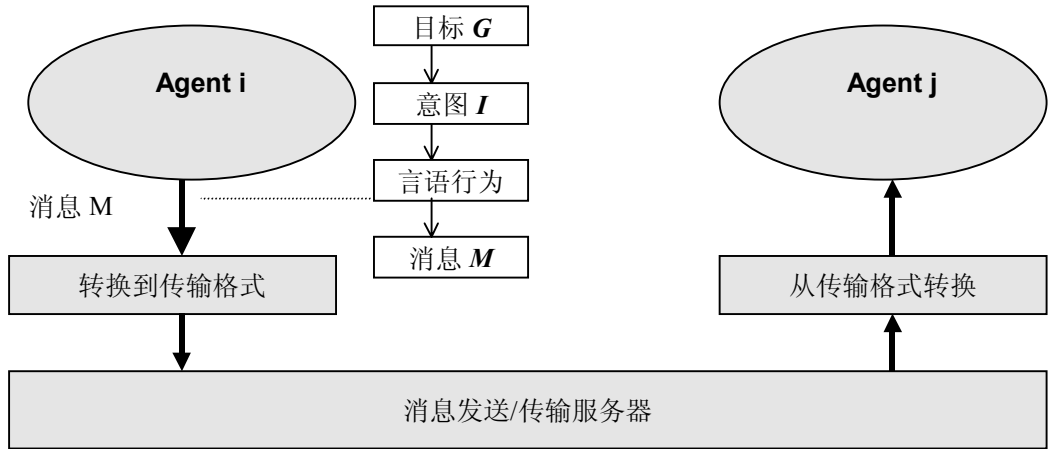


图 14.19 两个主体间消息传送

假设，主体 i 有以下心智状态：目标或者目的 G，意图 I。为了满足 G，主体采用一个具体意图 I。G 和 I 可以同样地用清楚的语言编码成一个 BDI 主体的心智结构，或者含蓄地在调用堆栈里，编程为一个简单的 JAVA 或数据库主体。

假设 i 不能由自己实现意图，问题就成为哪一种消息或者哪一组消息可以发送给其它主体帮助或导致意图 I 得到满足。如果主体 I 的行为是理性的，它将不会向一个不能满足意图的主体发送消息去完成目标。例如，如果 Harry 想要野餐，就会产生一个目标寻找哪儿合适，然后产生一个意图询问天气，他或她将不会糊涂地问 Sally “今天你买 Acme 的股票没有？”。从 Harry 看，不管 Sally 说什么，都不会对判断今天是否下雨有作用。接着这个例子，如果 Harry 正常行为，将问 Sally：“今天是否有雨？”他所做的是希望满足他的意图，达到他的目标（假设 Harry 认为 Sally 知道答案。）Harry 这样推理，问 Sally 的结果是 Sally 能告诉他，这样做一个请求完成意图。现在已经提出一个问题，Harry 能真正认为他迟早能知道是否有雨？Harry 可以认为 Sally 知道他不知道的事，并且她知道他正在询问她。但是，仅仅在提问的基础上，Harry 不能认为 Sally 能告诉他天气情况，因为她是独立的，可能正忙。

总之，一个主体通过与其它主体通信清楚地或者含蓄地（通过软件的结构）规划，用以最后达到目标，即发送消息给其它主体和从其它主体接收消息。主体将根据对于目标的动作期望值或者推理效果的适当依据选择动作。但是推理效果在发送消息以后并不能保证一定产生。

2. 语义语言 SL

语义语言 SL 是定义 FIPA ACL 语义学的形式语言。在 SL 中，逻辑命题表示成一个心智状态和动作的逻辑，使用具有恒等式的一阶模态语言。所用形式化的元素表示如下：

- $p, p_1 \dots$ 表示命题的闭合公式，
- ψ 和 ϕ 公式模式，代表任意闭合命题
- i 和 j 表示主体的模式变量，
- $\models \phi$ 表示 ϕ 有效。

主体的心智模型是基于三种原语状态：信念，不确定和选择（或者某种程度的目标）。它们分别用算子 B, U 和 C 表示。算子可这样读：

- $B_i p$ “i 相信 P”

- U_{ip} “ i 认为 P 不确定, 但认为 p 多过 $\neg p$ ”
- C_{ip} “ i 期望 P 现在正确”

算子 B 的逻辑模型具有固定领域原理的 KD45 可能世界语义学 Kripke 结构。

为了动作推理, 论域包括单个对象、主体和事件序列。一个序列可能只有一个事件。事件也可能为空。语言所涉及项包括全部事件序列。

关于复杂规划, 事件 (动作) 可以组合为动作表达式:

- $a_1; a_2$ 表示 a_2 接着 a_1 的序列
- $a_1|a_2$ 是非确定性的选择, a_1 或者 a_2 发生, 但不是都发生。

动作表达式可以用 a 表示。

算子 Feasible, Done 和 Agent 用于动作推理, 说明如下:

- Feasible(a, p) 表示 a 可能发生, 并且如果发生, p 将随后为真
- Done(a, p) 表示 a 刚发生, 之前 p 刚为真
- Agent(i, a) 表示执行主体 i 将执行动作 a
- Single(a) 表示 a 动作表达式而不是一个序列。任何单个动作都是 Single。组合动作 $a; b$ 不是 Single。组合动作 $a|b$ 为 Single 当且仅当 a 和 b 都为 Single。

持久目标的概念定义来源于信念、选择和事件。主体 i 有持久目标 p , 当 i 有目标 p , 并且一直追求目标 p 直到目标实现或者认为目标不可达到。意图定义为促使主体动作的一个持久目标。符号 PG_{ip} 和 I_{ip} 分别表示 i 有持久目标 p 和 i 有产生 p 的意图。I 的定义中意图必须生成一个规划过程。

嵌入心智状态或动作算子是没有限制的。例如: $U_i B_j I_j \text{Done}(a, B_i p)$ 表示主体 i 相信主体 j 认为 i 有这样的意图: 动作 a 能在 i 必须相信 p 时发生。

提出的逻辑的一个基本特性是模型化的主体与它自己的心智状态是完全相符的。正常时, 以下公式是有效的。

$$|= \phi \Leftrightarrow B_i \phi$$

其中 ϕ 是由表示主体 i 的心智状态的模态算子管理。

14.6 协调和协作

14.6.1 引言

在计算机科学领域, 最具有挑战性的目标之一就是如何建立能够在一起工作的计算机系统[128]。随着计算机系统越来越复杂, 将智能主体集成起来则更具挑战性。而主体间的协作是保证系统能在一起共同工作的关键。另外, 主体间的协作也是多主体系统与其它相关研究领域 (如分布式计算、面向对象的系统、专家系统等) 区别开来的关键性概念之一[50]。协调与协作是多主体研究的核心问题之一, 因为以自主的智能主体为中心, 使多主体的知识、愿望、意图、规划、行动协调, 以至达到协作是多主体的主要目标。

协调是指一组智能主体完成一些集体活动时相互作用的性质。协调是对环境的适应。在这个环境中存在多个主体并且都在执行某个动作。协调一般是改变主体的意图, 协调的原因是由于其它主体的意图存在。协作是非对抗的主体之间保持行为协调的一个特例。多主体是以人类社会为范例进行研究的。在人类社会中, 人与人的交互无处不在。人类交互一般在纯冲突和无冲突之间。同样, 在开放、动态的多主体环境下, 具有不同目标的多个主体必须对其目标、资源的使用进行协调。例如, 在出现资源冲突时, 若没有很好的协调, 就有可能出

现死锁。而在另一种情况下，即单个主体无法独立完成目标，需要其它主体的帮助，这时就需要协作。

在多主体系统中，协作不仅能提高单个主体以及由多个主体所形成的系统的整体行为的性能，增强主体及主体系统解决问题的能力，还能使系统具有更好的灵活性。通过协作使多主体系统能解决更多的实际问题，拓宽应用。尽管对单个主体来说，它只关注自身的需求和目标，因而其设计和实现可以独立于其它主体。但在多主体系统中，主体不是孤立存在的。即使由遵循某些社会规则的主体所构成的多主体系统中[Shoham 1995]，主体的行为必须满足某些预定的社会规范，而不能为所欲为。主体间的这种相互依赖关系使得主体间的交互以及协作方式对主体的设计和实现具有相当大的制约性，基于不同的交互及协作机制，多主体系统中的主体的实现方式将各不相同。因此可以说，研究主体间的协作是研究和开发基于主体的智能系统的必然要求。

在现阶段，针对主体协作的研究大体上可分为两类，一类将其它领域（如博弈论 Game Theory、经典力学理论等）研究多实体行为的方法和技术用于主体协作的研究[Kraus 1997]；而另一类则从主体的目标、意图、规划等心智态度出发来研究多主体间的协作，如 FA/C 模型[Leesser 1991]、联合意图框架(Joint Intention Framework)[Levesque 1990]、共享规划[Grosz 1996]等。但前一类方法的适用范围远不如后一类广，所运用的各种理论只适用于特定的协作环境下，而一旦环境发生变化，如主体的个数、类型、及主体间的交互关系与该理论所适用的情形不一致时，基于该理论的协作机制就失去了其存在的优势。而后一类方法则较偏重于问题的规划与求解，并且它们所假定的协作过程差异显著。有的是先找协作伙伴再规划求解，有的先对问题进行规划然后由主体按照该规划采取协作性的行动，而有的则在主体自主行动的过程中，进行部分全局规划(Partial Global Planning, PGP)来调整自己的行为以达到协作的目标[Lesser 1991]。后两种方式的协作显得较松散，缺乏必要的意外处理机制及手段，并且在主体间还必须存在共享的协作规划；而前一种方式的协作的不确定性较大，其协作规划会受到协作团体的影响和制约。

在多主体系统中，主体是自主的，多个主体的知识、愿望、意图和行为等往往各不相同，对多个主体的共同工作进行协调，是多主体系统的问题求解能力和效率得以保障的必要条件。包括组织理论、政治学、社会学、社会心理学、人类学、法律学、以及经济学等在内的多个学科领域都对协调进行了研究，许多研究成果已经应用到多主体系统中。多主体系统中的协调是指多个主体为了以一致、和谐的方式工作而进行交互的过程。进行协调是希望避免主体之间的死锁和活锁。死锁指多个主体无法进行各自的下一步动作；活锁是指多个主体不断工作却无任何进展的状态。多主体之间的协调已经有很多方法，大致归纳如下：

组织结构化 (Organizational Structuring)

合同 (Contracting)

多主体规划 (Multi-agent Planning)

协商 (Negotiation)

从社会心理学的角度看，多主体之间的协作情形大致可分为：

- 协作型：同时将自己的利益放在第二位。
- 自私型：同时将协作放在第二位。
- 完全自私型：不考虑任何协作。
- 完全协作型：不考虑自身利益。
- 协作与自私相混合型。

主体的交互有两种关系：负关系和正关系。负关系导致冲突，对于冲突的消解构成协调。解构成协调。正关系表示主体的规划有重叠部分，或某个主体具有其它主体不具备的能力，各主体可以通过协作获得帮助。分布式人工智能对协调与协作的研究主要在无目标冲突的情况

下互相帮助,实现目标。这种研究适用于分布式问题求解。20 世纪 80 年代中期,Rosenschein 在其博士论文中对于主体在目标有冲突情况下的交互进行了深入的研究,运用对策论,建立了“理性主体(rational agent)”交互的静态模型[Rosenschein 1986],成为多主体协调与协作问题的形式化理论基础。此后,许多学者运用对策论对于多主体的协商、规划、协调进行了形式化研究。这些研究都是在主体目标矛盾的前提下,如何通过建立对方模型或通过协商协调各自行为,或通过协作实现共同目标。有的研究考虑时间偏好,有的面向开放环境。

麻萨诸塞大学在强化 FA/C 与 PGP 方法的基础上,采用元级通信的方法,协调主体的计算。MacIntosh 运用启发式方法将协作引入机器定理证明。Sycara 以劳资谈判为背景研究协商问题[Sycara 1996]。该方法使用启发式与约束满足技术解决分布式搜索问题,使用异步回溯恢复不一致的搜索决策。其缺点是需要一个仲裁器解决冲突。Conry 研究了多目标、多资源条件下的多步协商。Hewitt 提出开放分布式人工智能系统的思想,对 Rosenschein 的静态交互模型提出挑战。现实世界是开放的、动态的,协调与协作也应是开放的、动态的。计算生态学认为,主体在开放的、动态的环境中不一定具备很强的推理能力,而可以通过不断的交互,逐步协调与环境以及各自之间的关系,使整个系统体现一种进化能力,类似于生态系统。在 BDI 模型中则强调交互作用中主体信念、愿望和意图的理性平衡。

Shohan 等提出为人工主体社会规定一套法规,要求每个主体必须遵守,并且相信别的主体也会相信。这些法规一方面会限制每个主体所能采取的行动,另一方面也可以确保其他主体会有什么样的行为方式,从而保证本主体行为的可实现性。Decker 等提出一种用于分布式传感器网络的动态协调算法[Decker 1995],各主体按照一个统一的标准动态修改本主体负责的传感器区域,可以自动达成各区域边界的协调。这种动态重构处理区域可以在整个系统内实现负载均衡,性能优于静态分区算法,尤其是可以降低性能的波动性,更适合于动态实时情形。

在多主体规划系统中,可以通过规定各种行为与目标的相关性而达成行为序列的自发协调。德国慕尼黑技术大学的 Weiß 通过分布式学习,形成对某个问题的一组行为与目标的相关值。自发协调的另一种方法是 Kosoresow 提出的 Markov 过程。在主体目标和偏好相容的情况下,Markov 过程是一种快速的概率性的协调方法。将 Markov 过程作为主体的推理机制,可以分析主体交互过程的收敛性和平均收敛时间。当主体目标和偏好不相容时,可以在某个时间限制点检测出不相容,并提交给更高层的协调协议。为了描述协同工作的一群主体行为,需要用共同意图将群体成员的行为联结起来。Jennings 等采用共同责任概念,强调意图作为行为控制器的作用,规定各主体在协作问题求解中应该如何行动。这种共同责任可以为系统结构设计提供功能指导,为监控问题求解提供标准,为异常处理提供准则。他们在 GRATE*中实现了这种责任方法。

14.6.2 合同网

1980 年 Smith 在分布式问题求解中提出了一种合同网协议(contact net protocol)[Smith 1980]。后来这种协议广泛用在多主体系统的协调中。主体之间通信经常建立在约定的消息格式上。实际的合同网系统基于合同网协议提供一种合同协议,规定任务指派和有关主体的角色。图 14.20 给出了合同网系统中结点的结构。

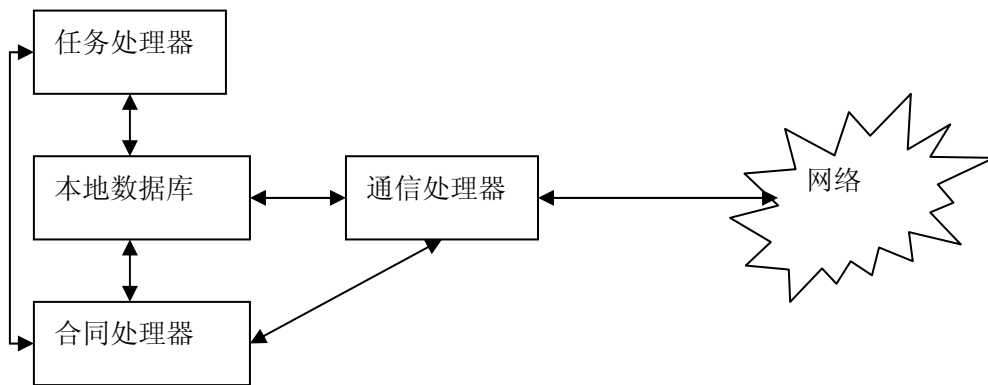


图 14.20 合同网结点结构

本地数据库包括与结点有关的知识库、协作协商当前状态和问题求解过程的信息。另外三个部件利用本地知识库执行它们的任务。通信处理器与其它结点进行通信，结点仅仅通过该部件直接和网络相接。特别是通信处理器应该理解消息的发送和接收。

合同处理器判断投标所提供的任务，发送应用和完成合同。它也分析和解释到达的消息。最后，合同处理器执行全部结点的协调。任务处理器的任务是实际处理任务赋予它的处理和求解。它从合同处理器接受所要求解的任务，利用本地数据库进行求解，并将结果送到合同处理器。

合同网工作时，将任务分成一系列子问题。有一个特定的结点称作管理器，它了解子问题的任务（见图 14.21）

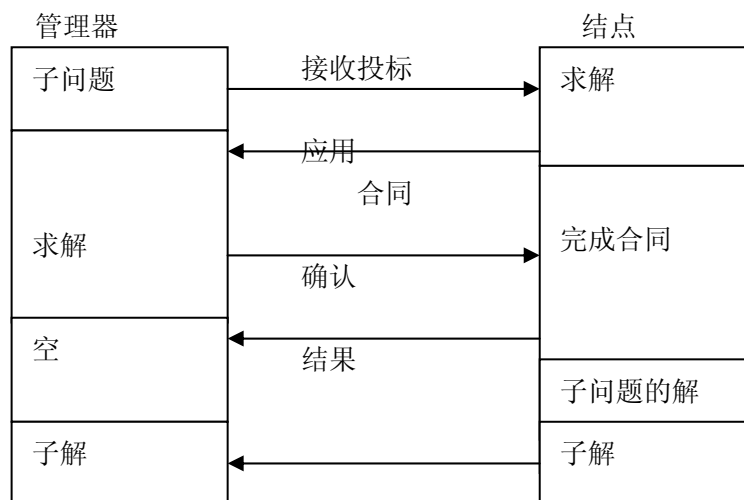


图 14.21 合同网系统中合同协商过程

管理器提供投标,即要解而尚未求解的子问题合同。它使用合同协议定义的消息结构，例如：

TO:	All nodes
FROM:	Manager
TYPE:	Task bid announcement
ContractID:	xx-yy-zz
Task Abstraction:	<description of the problem>
Eligibility Specification:	<list of the minimum requirements>
Bid Specification:	<description of the required application>

information>

Expiration time <latest possible application time>

标书对所有主体都是开放的，通过合同处理器进行求解。使用本地数据库求解当前可用的资源和主体知识。合同处理器决定该公布的任务申请是不是要做。如果要做，它将按下面结构通知管理器：

TO: Manager
FROM: Node X
TYPE: Application
ContractID: xx-yy-zz
Node Abstraction: <description of the node's capabilities>

管理器必须选择结点，全部应用中它最适合于所给的合同。它访问具体的求解知识和方法，选择最好成绩，将合同有关的子问题求解任务交给它。根据合同消息管理器指派合同如下：

TO: Node X
FROM: Manager
TYPE: Contract
ContractID: xx-yy-xx
Task Specification: <description of the subproblem>

通信结点发送确认消息到管理器，以规定的形式确认接受合同。当问题求解阶段完成，已解的问题传给管理器。承诺的结点完全负责子问题的求解，即完成合同。合同网系统纯粹是任务分布。结点不接受其它结点当前状态任何信息。如果结点晚些认为所安排的任务超过它的能力和资源，那么可以进一步划分子问题，分配子合同到其它结点。这时，它用作管理器角色，提交子问题标书。形成分层任务结构，每个结点可以同时是管理器、投标申请者和合同成员。

原来合同网系统作了一些扩充，影响协商的过程。其中之一是公布标书。所有结点都可以参加投标，这要求通信频繁和丰富的资源。管理器必须评价大量的投标书，使用大量的资源。管理器很重的负载可能是公共的投标请求造成的。首先，它要有能力只通知投标中的一小部分。可以想象，管理器具有各个结点能力的具体知识，那么它就能粗略估计处理子问题的可能的候选结点。第二点，公共投标请求完全可以取消。如果未解的子问题可以采用以前求解问题的方法构建，那么，管理器可以直接与过去求解问题的结点联系，如果资源可以利用，签订合同。另外，结点自己也可以投标。这种情况下，管理器许多开放的投标只是调查新的任务。投标请求只是在没有找到合适的投标者时需要。

合同网系统扩充的第二方面是影响实际合同的指派。原协议中，管理器在指派合同后要等待接受有关结点的信息。当确认信息到来之前，管理器不知道结点是否接受合同。结点投标后并未形成合同，没有建立合同约束。建议的扩充是将合同约束建立移到协商的早期。例如，当一个结点投标时，可以提供后面接受承诺可能的条款。与此相关，接受可能性不是简单的接受或拒绝，而可以带有一些参数或条件。合同确认的最大期限是进一步的扩充。如果在规定期限内结点没有确认合同，那么管理器将中断合同。合同处理器也可以发送信息，避免管理器等待太长的时间，在最长间隔之前，管理器就可以重新指派合同。

14.6.3 部分全局规划

部分全局规划(Partial Global Planning, PGP)方法最重要的特点是多主体系统中工作

时每个主体的能力是给定的, 收集当前状态的信息, 其它主体达到的目标。主体可以使用知识优化它们的任务。PGP 提供了一种灵活的概念, 协调分布式问题求解部件。

使用 PGP 的基本条件是需要几个分布式主体为整个问题求解工作。一个主体作为 PGP 的一部分, 考虑同组其它主体的动作和关系来形成自己的结论。这种知识称作部分部分全局规划, 它反映求解全局问题的一个主体决定的部分规划知识。图 14.22 给出一个例子, 说明 PGP 系统的基本工作原理。两个主体分别工作两个子问题(A 和 B). 每个主体发送信息到它的合作者主体 1 通知

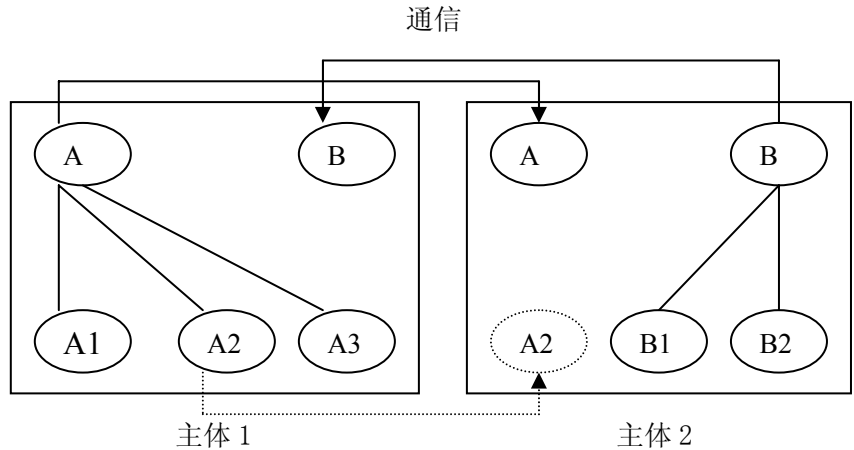


图 14.22 部分全局规划例示

主体 2 它当前工作子问题 A。同样, 主体 2 通知它工作子问题 B。每个主体可以利用该信息知道合作者的情况。例如, 主体 1 知道它的子问题 A2 取决于主体 2 的子问题 B, 它可以通知主体 2。PGP 处理过程可以分为四步:

- (1) 每个主体创建局部规划
- (2) 主体之间通信和交换规划
- (3) 创建部分局部规划
- (4) 修改和优化部分局部规划。

协调过程开始之前, 每个主体必须创建局部规划, 求解指派的任务。每个局部规划至少要有不同层次的两级。整体结构包含求解问题最重要的步骤, 反映主体长期求解问题规划; 详细结构包括每个具体子问题的详细信息。

一旦局部规划完成, 第二步, 主体之间交换知识。每个主体必须具有一定数量的专门组织的知识, 它可以决定在问题求解中其它主体的角色, 什么信息感兴趣。元级组织决定主体层次。一旦主体收到其它主体的规划信息, 它必须以 PGP 形式组织。检查有否新的信息包括对它内部规划的依赖关系, 然后, 将这些相关的子规划分组成为逻辑部件。部分全局规划由下列部件构成。

- (1) 目标。目标包含 PGP 基本信息, 包括存在的理由、长期目标、与其它规划比较的优先级。
- (2) 规划活动图。规划活动图包括其它主体的任务, 它们当前的状态, 详细计划, 期望的结果和有关尝试。
- (3) 求解构造图。求解构造图包括主体之间怎样进行信息通信和彼此协作。主体发送的规划的具体大小和时间特别重要。
- (4) 状态。状态包括报告 PGP 全部重要信息, 例如其它主体所接受的规划、接受的时间标志。

PGP 规划器的核心是分析其它主体送来的信息, 决定几个主体是不是工作于相同目标。PGP 规划器以规划活动图方式集成知识。它也预测其它主体未来的行为和结果。规划活动图形成

主体未来工作的基础。规划图用来把局部规划与新的知识比价,创建修改的局部规划。这时也创建求解构造图。PGP 规划器产生修改的局部规划作为最终结果。具体规划送到具体主体后,规划器要使用整个系统当前的知识进行优化和充实细节。

使用局部规划器有许多优点。最重要的好处是高度动态的系统行为。所有规划任何时候都能适应新的环境求解,整个系统具有高度灵活性和有效性。PGP 尽可能识别原规划的改变。必须按时地将所作的修改送到其它结点,因为修改会影响它们的工作。如果任何细小的修改都要通知,那么就会加重整个系统的负担。因此可以允许存在小的不一致性,仅仅通信重要的修改。系统开发者必须开发这样的容错系统,并规定一个阈值来指示修改是重要的。第二个好处是有效性和避免冗余。两个或多个主体工作相似或相同的问题,将马上登记它们整个 PGP,然后在它们中重新指派或构建。工作的指派非常有效。

当然,原来的 PGP 模型有许多限制,特别在多主体系统中异构主体不同问题求解。怎样处理动态主体重复改变求解策略?能否工作在实时环境下,即要在规定时间内完成具体任务?主体之间怎样进行协商?有一些方案对 PGP 模型进行扩充,例如通用部分全局规划 GPGP (General Partial Global Planning, GPGP) 提供新的部件来满足这种要求[44]。

Osawa 以机器人协作为背景,提出一种在开放环境下(知识不完备、不准确、能力不同)构造协作规划的方法[Osawa 1993]。在这一方法中,理性主体使用“多世界模型”构成不完全的单主体规划,再根据基于效用的模型协商平衡费用,从而构成协作规划。这一过程可以描述为:

- (1) 需求者(requestor)向公告板主体发送需求建议 RFP
- (2) 空闲主体向公告板主体申请一个 RFP
- (3) 公告板将 RFP 发到提出申请的空闲主体
- (4) 空闲主体产生个体规划
- (5) 空闲主体将其规划发给需求者
- (6) 需求者调查协作的可能
- (7) 需求者发送协作奖励
- (8) 申请者组成协作规划

其效用值可用下式计算:

$$\text{utility}(a, g) = \text{worth}(a, g) - \text{cost}(\text{plan}(a, g))$$

效用的平均是协作的原则。尽管 Osawa 在一定程度上解决了开放环境中主体协作的问题,但将各主体效用简单相加再平均的方法仍然太弱,因为主体效用仅是主体本身对目标偏好的一种排序关系,不同主体效用一般不能用数值比较。

14.6.4 基于约束传播的规划

在多主体系统中,主体协调工作可以通过情景动作(situated action)和规划(planning)来实现。基于情景动作的方法强调主体和环境的交互以及紧耦合的感知—动作循环在主体行为中的作用。规划方法则利用某种搜索方法来确定主体要执行动作的顺序。相对于情景动作而言,规划可以保证主体行为的正确性,使主体更具有理性。因而,被广泛地应用于“慎思”主体中。

但是,由于缺乏对多主体系统中主体间协调、协作的考虑,目前大多数通用规划算法,如 UCPOP, GraphPlan, SNLP 等并不能被直接应用。在多主体环境下,主体间的动作是并行的,不由单一主体控制的;一个主体可以通过通讯和协商来试图改变其它主体的行为使系统更加协调,可以通过请求其它主体的协作来完成自身能力所不及的工作。由于经典的规划算法假

设主体自己是唯一可以改变其所在环境的，因而规划的结果不能保证不和其它主体冲突，并且主体的求解能力也受到了限制。另一方面，部分全局规划（Partial Global Planning）等动态环境下的分布式规划假设环境中的突发事件是不可预料的，并且主体的规划和执行交替进行。这种方法不能保证规划的可靠性和完备性。

一个规划，特别是偏序规划可以由下面规划步骤上的约束来确定：用时序约束（temporal constraint）来指定执行步骤间的时间先后顺序，用等价约束（codesignation constraint）来不完全指定执行的步骤。规划求解就是对规划逐步添加细化约束的过程。主体之间的冲突协调和主体间的协作也可以通过指定动作之间的时序和等价约束来实现。不同主体的动作之间的资源占用冲突和一个主体为其它主体的服务可以用主体动作间的因果链（causal link）来表示，并且通过加入主体动作之间的时序约束来解决。同时，主体间的协作动作，如两个主体合作抬起一张桌子，必须用单独的动作描述来表示。多主体环境下一个主体规划的正确性由真值条件（truth criterion）保证。我们提出了一种多主体并行偏序规划算法，每个主体并发地对其要完成的目标进行规划，当某个主体决定在规划中引进一个新的动作时，它通过和其他主体间的约束传播来实现多主体系统的协调[262]。可以证明分布式多主体并行偏序规划算法是可靠的。

1. 规划表示

在规划动作的描述上，STRIPS 表示在实现实用的规划器中限制过多，而完全采用情景演算又过于困难。1989 年 Pednault 提出了一种动作描述语言 ADL。我们扩充 ADL 来描述多主体系统中的规划动作，它的表示能力介于 STRIPS 和完全的一阶逻辑之间。

ADL 的语义基于描述世界状态的数学结构。一个动作 a 在 ADL 中用一个状态对 $\langle s, t \rangle$ 表示，其中，动作 a 在状态 s 被执行并且产生状态 t 。一个状态 s 和状态描述 ϕ 的联系用模型符号 $|=$ 表示，记作 $s \models \phi$ 。ADL 中的动作模板代表了一组可能的动作，用四组可选的子句来描述：Precond：前提条件；Add 和 Delete：状态 t 的关系 R 中被加入和删除的公式集合；Update：一组描述函数如何从 s 变到 t 的关系。当用 $R()$ 表示加入， $\neg R()$ 表示删除时，Add 和 Delete 可以被合并为 EFFECTS。但是本文限制前提条件为析取并且不使用 UPDATE。更形式化地，一个动作模板可以表示为动作步骤和它的效果。

定义 14.1 一个动作步骤 S 是一个四元组 $\langle \rho, \varepsilon, \beta, \gamma \rangle$ ，其中， ρ 是步骤的前提，是一组量化（quantified）的文字； ε 是步骤的效果集； β 是步骤的约束，是 ρ 中变量的一组等价约束或不等约束； γ 是步骤执行时要占用的资源，用对象的文字来表示。

用 ρ_S 、 ε_S 、 β_S 、 γ_S 表示在步骤 S 中的集合 ρ 、 ε 、 β 、 γ 。 β 中的等价约束应用于效果 ε 和前提 ρ 中的所有公式，为一组耦合对 (u, v) 或 $\neg(u, v)$ 。前者表示自由变量和常量，后者表示自由变量。 u 和 v 必须等价，即，它们可以合并到同一个合式公式中； $\neg(u, v)$ 表示不等价，即 u 和 v 不能被合并到任意的合式公式中。 β 规定了规划中变量和常量上的等价关系，

记作 \approx_β 。 γ_S 是动作步骤 S 所占有的资源。我们假设某一资源在动作执行时，必须为其所独占。所以，两个占用资源有交集的动作步骤不能同时被执行，否则会发生资源的冲突，必须进行协调。值得注意的是前提 ρ_S 和资源 γ_S 的不同，前提 ρ_S 要求在 S 的实例 A 执行前为真，即在状态 s 前为真；资源 γ_S 要求在 S 的实例 A 执行过程中为空闲，即在状态 s 和状态 t 之间为空闲。

定义 14.2 动作效果是一个三元组 $\langle \rho, \theta, \beta \rangle$ ，其中， ρ 是效果的前提条件， θ 是效果的后续条件，它们是一组量化的文字； β 是 ρ 和 θ 上自由变量上的等价约束。

例如，考虑对把一个积木 b 从位置 x 移到位置 y 的例子。设动作步骤为 move(b, x, y)。它可以描述为：

$\{\{on(b, y), clear(b), clear(y)\}, \{\{\phi, \{on(b, y), \neg on(b, x), clear(x)\}, \phi\}, \{(y \neq table), \neg clear(y), \phi\}\}, \{b \neq x, b \neq y, x \neq y\}, \phi\}$

ϕ 表示空集。它一种更可读的 Common Lisp 表示如下：

```
(define (operator move)
  : parameters (?b ?x ?y)
  : precondition (and (on ?b ?x) (clear ?b) (clear ?y)
    (≠ ?b ?x) (≠ ?b ?y) (≠ ?x ?y))
  : effect (and (on ?b ?y) (not (on ?b ?x)) (clear ?x)
    (when (≠ ?y table) (not (clear ?y)))))
```

一般来说，不同主体动作之间的关系可以用它们之间的约束来表示，如，主体 α 的动作 $A_{\alpha i}$ 的执行是另一个主体 β 的动作 $A_{\beta j}$ 执行的前提可以用这两个动作的因果链(定义见下节)表示。

在一个主体的内部表示中，这种约束关系表示为来自其它主体的承诺和主体自身动作和外部承诺之间的约束关系。一个主体答应为其它主体完成某件事时，它向其它主体提供一个承诺使其相信它一定会完成这件事。有了其它主体对某动作 A 的承诺，主体就可以在推理时相信 A 的有效性，并把 A 等同于自己的动作而利用 A 的结果或者避免其它动作和 A 冲突。另外，值得考虑的是在不同主体之间的合作动作。对于 n 个动作 A_1, A_2, \dots, A_n 和某一性质 p，如果 p

在 A_1, A_2, \dots, A_n 被并行执行后为真，但是 A_1, A_2, \dots, A_n 中任何一个动作没有被执行时都不为

真，则称 A_1, A_2, \dots, A_n 为合作动作 (joint-action)，p 为合作动作的结果之一。合作动作在

多主体系统中是常见和不可少的，例如，两个主体合作来抬起一张桌子。由于单个主体的动作不能反映合作动作的效果，一个合作动作不能用每个主体上的动作分量单独表示，它们必须用单独的动作模板来描述。同时，合作动作中，各个主体的动作分量 A_1, A_2, \dots, A_n 之间是时序等价的。如果它们是时序等价的，动作执行的时间等价由主体同步信息来保证。时序等价的定义如下：

定义 14.3 一个偏序集上 $\langle S, \leq \rangle$ 的时序等价关系定义为：对于任意 $s_1, s_2 \in S$ ， $s_1 \approx s_2$ 当

且仅当：对于任意 $t \in S$,

(1) 如果 $t \leq s_1$, 有 $t \leq s_2$; 如果 $s_1 \leq t$, 有 $s_2 \leq t$ 。

(2) 如果 $t \leq s_2$, 有 $t \leq s_1$; 如果 $s_2 \leq t$, 有 $s_1 \leq t$ 。

在偏序规划中,因果链(causal link)被用来显式地记录动作之间的依赖关系。根据 Austin Tate 和 Daniel Weld 的工作,因果链可定义为

定义 14.4 因果链是四元组 $\langle s_i, e, r, s_j \rangle$, 表示为 $s_i \xrightarrow{e,r} s_j$; 其中, r 是 s_j 的前提(或者效果的前提)之一, e 是 s_i 的效果之一。并且, 存在 $\exists q \in \theta_e$ 使 q 和 r 《归一》(unified)。

定义 14.5 规划是四元组 $\langle S, B, O, L \rangle$ 。 S 是一组动作步骤, 包括主体自身的动作和其他主体的动作承诺; B 是 S 上自由变量的 binding 约束; O 是一组偏序约束 $\{s_i \prec s_j \mid s_i, s_j \in S\}$; L 是因果链的集合。

定义 14.6 规划问题是四元组 $\langle \Lambda, I, U, \Gamma \rangle$ 。 Λ 是动作模板的集合; I 是一组表示初始情景的文字; Γ 是表示目标的量子子句集(quantified clauses); U 是 Λ, I, Γ 中变量的论域。

定义 14.7 目标规划。 对一个问题 $\alpha = \langle \Lambda, I, U, \Gamma \rangle$, 它的目标规划 $\text{g-plan}(\alpha)$ 是规划 $\langle S, B, O, L \rangle$ 。 其中, $S = \{s_0, s_\infty\}$, $O = \{s_0 \prec s_\infty\}$, B 和 L 为空集; 并且, $\varepsilon_{s_0} = I$, $\rho_{s_\infty} = \Gamma$, s_0 和 s_∞ 中的其他元素(element) 为空集。

2. 多主体环境下的规划问题

多主体的规划和单个主体的规划有许多不同之处。例如, 主体之间必须进行规划信息的交流来协调它们之间的规划冲突。但是, 从状态空间来考虑规划, 真值标准[4]依然适用。一个性质 p 在状态 s 下为真条件由下面的真值标准给出: 一个性质 p 在某一状态 s 下为真的充要条件是存在一个性质建立状态 $t \leq s$, 使性质 p 为真; 并且, 在性质 p 的建立状态 t 和状态 s 之间不存在使 p 不成立的终止状态。它可以形式化的表示为:

一个性质 p 在状态 s 为真 iff:

- (1) 存在一个状态 t 使 u 为真, 既 $\text{holds}(u, t)$ 。
- (2) $p = u$
- (3) $t \leq s$
- (4) 对于所有的状态 w 和性质 q 使得 $\text{holds}(\neg q, w)$

或者 (a) $p \neq q$

(b) $s \prec w$

(c) 对于各种变量的等价关系使 $p = q$, 存在一个状态 v , 其中

$\text{holds}(\neg r, v)$ 并且 $w \prec v \leq s$, $r = p$

其中, \prec, \leq 表示状态之间的时序关系, $=, \neq$ 表示性质之间的等价或不等价关系, $\text{holds}(v, t)$ 表示性质 v 在状态 t 时为真。作为一条定理, 真值条件的正确性已经被 Chapman 在[4]中证明了。利用此真值条件, 可以导出一种最小承诺(least commitment)的规划生成思想。按照

这个思想，规划的目标求解过程可以用真值条件解释为两个部分，一是性质的建立阶段，通过引入新的和利用已有的动作来实现；二是性质的维护阶段，把对已建立的性质有威胁进行升级或降级使性质在需要成立的时候得到保护。对每个性质的维护是通过因果链记录动作之间的依赖关系来保护引入的动作不会干涉其它动作的推理实现。

提供服务、目标冲突、资源冲突和合作动作是多主体规划中特有的问题。利用真值标准和上节中的规划表示，对它们在多主体系统中解决方案的讨论如下：

提供服务：主体 α 在推理时，需要使某一断言 p 为真，但是它不在主体 α 的任何动作 $A_{\alpha i}$ 的效果中。这时，主体 α 需要另一个主体 β 的动作 $A_{\beta j}$ (p 是 $A_{\beta j}$ 的效果之一) 被执行使 p 成立。如果主体 α 和 β 达成了服务提供的协议，主体 β 向主体 α 提交承诺 $\text{Comm}(A_{\beta j})$ ，并且在主体 α 和 β 的当前规划中建立因果链 $A_{\beta j} \xrightarrow{p} A_{\alpha i}$ 和偏序约束 $A_{\beta j} \prec A_{\alpha i}$ 。另外一种情况是，主体 α 的动作 $A_{\alpha i}$ 和主体 β 的动作 $A_{\beta j}$ 有重复的结果 p ，假设动作 $A_{\beta j}$ 被先加入了规划中， α 可以先请求 β 的服务，而不是自己采用动作 $A_{\alpha i}$ 。这样做的好处是避免了因为重复动作导致的对因果链的威胁和它带来的回拗问题。

目标冲突：一个动作 s 的目标是使性质 p 在时态 t 为真，如果另一个动作 s' 的效果是使性质 p 在时态 t 不可为真，那么就要按真值标准中的条件 (4) 进行性质的维护，既对 s' 进行升级或者降级。由于所有的主体共有一个初始动作 s_0 并且它们的终止动作 $\{s_\infty\}$ 是一个规划任务的子目标，同时，任意一个动作 s (除 s_0 和 $\{s_\infty\}$ 之外) 都要满足 $s_0 \prec s \prec \{s_\infty\}$ 。因而，如果在 $\{\rho_{s_\infty} = \Gamma\}$ 中存在一个性质 p 和它的否定，不可能对 s_∞ 进行升级或降级，则不可能得到一个无冲突的规划。

资源冲突：两个占用相同资源的动作 A_1 ， A_2 不能同时被执行，必须在它们之间加入时序约束 $A_1 \succ A_2$ 或者 $A_1 \prec A_2$ 。为了使规划器知道资源 r 在动作 s 时被占用，我们定义了一种伪因果链 $s \xrightarrow{\neg \text{Free}(r)} s$ 。并且把 $\text{Free}(r)$ 做为动作的前提之一。这样，资源冲突的动作之间就会建立时序约束，从而避免了资源冲突。

合作动作：多主体系统中，不同主体的动作可以被并发地执行。有两种可能的情况：某性质 p 在并发动作 $A_1 \parallel A_2$ 执行时被否定，而单独的动作 A_1 或 A_2 执行时成立；某性质 p 只有在并发动作 $A_1 \parallel A_2$ 执行时为真，而单独的动作 A_1 或 A_2 执行都不成立。第一种情况可以用因果链保护性质 p 使 A_1 和 A_2 不能被同时执行。对于第二种情况，由于从 A_1 和 A_2 单独的动作模板 S_1 和 S_2 中，规划器知道都不能了解并发执行 A_1 和 A_2 可以使性质 p 为真。因而，合作动作

$A_1 \parallel A_2$ 必须显式地用动作模板描述。合作动作必须是时序等价的。所以，对任何一个动作分量上的时序约束也要加到另一个分量上。

3 规划算法

多主体规划是一个或多个并发的主体进程同时对各自的目标在其规划空间的搜索。由于协调不同主体动作的需要，要在可能冲突的动作之间加入偏序约束。这些约束分布在不同的主体中，所以存在判断分布的偏序约束的一致性的问题。首先引进递增圈的定义。

定义 14.8 递增圈。一个偏序图是一个标记图 $\langle V, E \rangle$ ，其中， V 是变量结点的集合，边 E 是关系表达式， xRy 的集合， $x, y \in V$ ， R 为偏序关系等于或小于，记作 $=$ 或 $<$ ，图中的一条递增路径是变量结点的序列使得对任意是图中一条边，如果，则称该路径是一个递增圈。

显然，下面命题成立，

命题 14.1 偏序图蕴涵不一致性，当且仅当偏序图中存在一个递增圈，而且其中某两个变量结点（可能相同）为 $<$ 关系。

命题 14.2 一个一致的规划中，如果存在一个递增圈，则圈中的所有变量都是一个合作动作的投影。

规划算法中，动作间的偏序关系是随着规划的进行而逐步被指定的。即，总是把新的约束 $A_i < A_j$ 加入一组原本一致的约束集中或者从一致的约束集中减去约束。此时，下面命题成立。

图 5.5 分布多个主体中的递增圈。(a. 递增圈在一个主体内；b. 递增圈存在于两个主体中；c. 多个主体中的递增圈)

命题 14.3 在规划 $\langle S, B, O, L \rangle$ 中，如果由于加入约束 $A_i < A_j$ 而导致 O 不一致，则 (1) O 中存在递增圈，(2) $A_i < A_j$ 一定是递增圈中的一条边。

命题 14.4 在一致的规划 $\langle S, B, O, L \rangle$ 中减去约束 $A_i < A_j$ ，规划仍然一致。

在多主体规划中，约束可能是不同主体动作之间的偏序关系，并且是分布在各个主体内部的。一个递增圈有可能存在于一个主体内部，两个或多个主体之间。发现多个主体之间的递增圈的存在需要在主体之间传递偏序信息，定理 14.1 说明了递增圈的计算可以简化为动作 $A_i < A_j$ 或 $A_j < A_i$ 传递闭包的计算。由它可以得出一种其在分布的主体中的计算方法。当主体 α 要在它的规划中加入偏序约束 $A_i < A_j$ 时，它首先计算 A_i 在本地规划中 O_α 上 $<$ 传递闭包 $T_\alpha^<(A_i)$ ，如果 $A_j \in T_\alpha^<(A_i)$ ，则可知 O_α 是不一致的；否则要计算 A_i 的全局传递闭包。

定理 14.1 如果在一致的规划中加入约束 $A_i < A_j$ 后，有 $A_i \in T_\alpha^<(A_j)$ 或者 $A_j \in T_\alpha^<(A_i)$ ，其中， $T_\alpha^<(A_i)$ 是 A_i 的 $<$ 传递闭包，则规划不一致。

算法 14.1 (分布偏序约束一致性判断): $\text{calculate_transition_closure}(\alpha, A_i)$

输入：规划 $\langle S, B, O, L \rangle$ ，约束 $A_i \prec A_j$ ，

输出：传递闭包 $T_{<}(A_i)$ 。

BEGIN

发送信息通知所有主体把其规划中的时序约束 0 都设为只读；

计算动作 A_i 在主体 α 中的局部传递闭包 $T_{<}^{\alpha}(A_i)$ ；

$T_{<}(A_i) = T_{<}^{\alpha}(A_i)$ ；

设 $SET(agent_n)$ 为 $T_{<}(A_i)$ 中主体 $agent_n$ 的动作。

对于每一个非空的 $SET(agent_n)$ ，启动一个新的线程

begin

$T_{<}(A_i) = T_{<}(A_i) + \text{request } (\text{calculate_transition_closure}(agent_n,$

$SET(agent_n))$ ；

return.

end

当所有的子线程返回后，return $T_{<}(A_i)$ ；

通知所有的主体把约束退出只读状态。

END

算法 14.1 中，主体之间的合作包括两个阶段：对各个主体的任务分配和主体的子目标规划。任务分配把多主体系统的全局目标分解并分配给系统内的一个或一组主体来执行。这里侧重于研究后一个问题，即，开始规划时每个主体都有了自己的目标，它可能是被分配的或者是主体为了满足自己利益而生成的。另外，对多主体系统和其中的主体，假设：所有动作的效果都是这个工作和动作执行时系统状态的确定性函数。

主体拥有关于自身动作，其他主体的动作和系统初始状态的全部知识。

系统的变化只能有主体的动作引起。

不论主体能否成功地做出规划，它都维持在规划阶段同其他主体达成的动作承诺，使其它主体的规划可以继续进行。

主体之间的通信满足通信语言 KQML 规范中的消息传输约定。特别是，通讯要求无延迟和对同一目标消息是按发送的顺序到达的。

主体具有如图 14.23 所示的结构。由于主体之间进行协商，判断约束一致性的需要，在每个主体内有三个独立的线程运行，它们是：

规划线程：负责执行算法 14.2 来求解自身的规划问题。

约束维护线程：计算动作在自身动作集 S 上的局部传递闭包，判断分布的偏序规划是否一致。

通讯和协商线程：负责主体之间的动作约束传递和进行动作协作和协调。

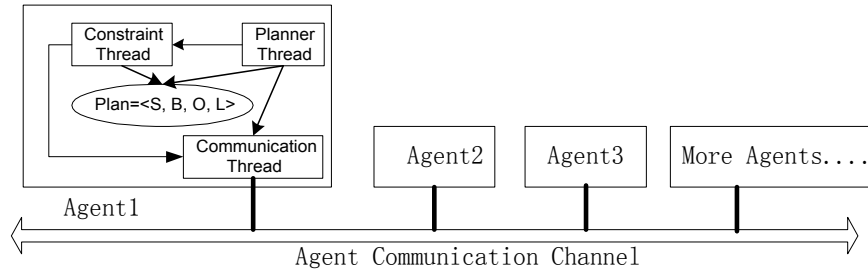


图 14.23 规划主体的内部结构

每个主体可以按算法 14.2 对自己的子目标进行规划求解。在算法 14.2 中，主体按照 UCPOP 的框架进行规划的建立和维护；当需要从其他主体请求服务和协作以及进行分布偏序约束一致性判断时，主体之间要进行消息的通讯。

算法 14.2 $\langle \langle S, B, O, L \rangle, G, \Lambda \rangle$

终止条件： 如果 G 为空，则返回 $\langle S, B, O, L \rangle$ 。

目标消解： 从 G 中取出一个目标 $\langle Q, A_c \rangle$ ，

如果 Q 是 Q_i 的合取，则把每个 $\langle Q_i, A_c \rangle$ 加入 G ，go to 2；

如果 Q 是 Q_i 的析取，则非确定地选择一个 Q_k ，并把它加入 G ，go to 2；

如果 Q 是一个文字，并且在 L 中存在一个链 $A_p \xrightarrow{-Q} A_c$ ，返回 failure。

选择操作算子： 非确定性地选择一个在 S 中存在的动作，或者从 Λ 选择一个新的动作 A_p ，其存在效果 e 和一个全称子句 $p \in T(\theta_e)$ ，并且 $MGU(Q, p) \neq \perp$ (Q 和 p 存在一个最一般的归结)；如果 Λ 中不存在满足条件的动作并且存在一个主体 β 其动作 A_{β} 满足条件，向主体 β 发出请求；如果存在合作动作满足条件，向所有合作动作的执行者发出请求；被请求的主体在其规划 P_{β} 中执行步骤 4 和 5，如果成功， β 返回 $Comm(A_{\beta})$ ，把合作动作或 $Comm(A_{\beta})$ 加入 S ，go to 2；否则， β 返回 failure。

使新动作有效： 令 $S' = S$ ， $G' = G$ ，如果 $A_p \notin S$ ，则加 A_p 到 S' 中，并把 $\langle preconds(A_p) \setminus MGU(Q, R, B), A_p \rangle$ 加入到 G' ，把 $non-cd-constraints(A_p)$ 加入到 B' 中。

因果链的保护： 对每个因果链 $l = A_i \xrightarrow{p} A_j$ 和每一个可能威胁到 l 的动作 A_t ，从下面三中方案中选一（如果没有选择，则返回失败）

(a) 升级：对 O 进行分布偏序约束一致性判断，如果约束一致， $O' = O \cup \{A_j < A_t\}$

(b) 降级：对 O 进行分布偏序约束一致性判断，如果约束一致， $O' = O \cup \{A_i < A_j\}$

(c) 面对：如果 A_i 所威胁的 l 为条件结论，令其条件为 S ，结论是 R ，则加入 $\langle \neg S \setminus MGU(P, R), A_i \rangle$ 到 G' 中。

递归调用：如果 B 不一致，返回 failure；否则调用 $(\langle S', B', O', L' \rangle, G, \Lambda)$ 。

算法 14.2 是可靠的，但却不是完备的。所谓可靠性，是指对于一个规划问题，如果主体采用的规划算法找到了一个规划，那么它是规划问题的解，下面的定理 14.2 给出了对可靠性的简单证明；算法 14.2 不是完备的，是因为主体在其规划空间上的搜索和其他主体搜索过程有时序和等价约束关系，所以，主体的搜索空间受到了限制。特别是，为了不使并行规划过于复杂和冗长，我们没有提供一种机制使一个主体在回朔后找不到解时，可以请求其他主体也回朔或者放松约束使请求主体当前时刻的解空间扩张仍可以继续进行搜索。

定理 14.2 (可靠性)：对于一个采用算法 14.2 的主体来说， $\alpha = \langle \Lambda, I, U, \Gamma \rangle$ 是它的规划问题。如果算法 14.2 返回一个规划 P ，则 P 是 α 的一个解。

证明概略：依照 Pednault 的因果关系定理和 Chapman 的真值标准，算法 14.2 的每一次迭代前后，规划都满足循环不变性（如果 G 中的子目标都被规划 P 满足，则 P 是 α 的一个解）。由于算法开始和终止时，算法的循环不变性都满足；并且，当算法终止时， G 为空，没有其他条件限制 P 是规划的解。所以算法 14.2 返回的规划 P 是 α 的一个解。

14.6.5 基于生态学的协作

80 年代末，在计算机中出现了一个崭新的学科——计算生态学 (the ecology of computation)。计算生态学是研究关于开放系统中决定计算结点的行为与资源使用的交互过程的学科。它摒弃了封闭、静止地处理问题的传统算法，将世界看作是开放的、进化的、并发的，通过多种协作处理问题的“生态系统”(ecosystem)加以研究。它的进展与开放信息系统的研究息息相关。

分布式计算系统具有类似于社会的、生物界组织形式的特征。这类开放系统与目前的计算机系统有很大差别，它们对于复杂的任务进行异步的计算，它们的结点可以在内部结构陌生的其他机器上产生进程。这些结点能根据不完备的知识与不完整的、经常迟到的信息做出局部决策。整个系统不存在中心控制，而是通过各结点的交互，协作解决问题。所有这些特点构成了一个并发的组合体，它们的交互、策略以及对资源的竞争类似纯粹的生态学。Hewitt 提出了开放信息系统的概念 [Hewitt 1991]，他认为不完备的知识、异步的计算以及不一致的数据是开放计算系统所无法避免的。人类社会，特别是科学界在面临同样的问题时能够成功地通过协作加以解决。

计算生态学将计算系统看作是一个生态系统，它引进了许多生物的机制，如变异 (mutation) 即物种的变化。这些变化导致生命基因的改变，从而形成物种的多样性，增强了适应环境的能力。这类变异策略成为人工智能系统提高其自身能力的一种方法。Lenat 与 Brown 成功地将变异机制引入他们的 AM 与 Eurisko 系统中，通过小型 Lisp 程序的语法变异发现数学概念。他们认为未来成功的系统应该是一系列进化的、自组织的符号知识结构的“社会”系统。

Miller 与 Drexler 讨论了一些进化的模型,如生物生态系统、商品市场,并指出它们与计算生态系统的异同。他们认为,一个直接计算市场是最理想的系统模型。

由于不完备知识与迟到的信息是计算生态系统的内在特征,在这些限制下系统的动态行为的研究是非常重要的。Huberman 与 Hogg 提出并分析了动态游戏的过程,指出当各进程为完成计算任务进行交互时,若有多个可选择策略,动态渐近过程可能变为非线性振荡与混沌。这表明在计算生态系统中进化的稳定策略有可能不存在。他们同时还讨论了可能存在的普遍规律以及合作在系统中的重要性,并且与生物生态系统和人类组织进行了比较。与动态理论相对应,Rosenschein 与 Genesereth 提出用静态对策理论解决具有不同目标的计算结点的潜在冲突问题。

目前,著名的生态系统模型有生物生态系统模型、物种进化模型、经济模型、科学团体的社会模型。大型生态系统的智能,超过任何个体智能。

1. 生物生态模型

这是最著名的生态系统,具有典型的进化特征和层次性。这种特性反映在“食物链”中。对于复杂的生物生态系统而言,各物种组成了紧密相连的网络——食物网。这个系统的主要角色是捕食者与被捕食者。生命依赖于生命,共同进化,由小的生态环境组成大的生态系统。

2. 物种进化模型

物种进化的“复制者”是基因。从门德尔的植物遗传研究到现代遗传学的成果,都说明了在物种进化过程中,基因的组合与变异起着关键作用。在一个物种的某一群体中基因的集合称为基因池。生物组织是基因的载体。如果环境变化,选择的机制就会改变。这种变化必然引起基因池的变化。特定种群的基因变化称为基因流。一个物种总是不断地经历隔绝、基因流动、变化的循环。开始时,一组地理上隔绝的群体自己孤立地发展,基因在内部快速地流动。随着开放,通过交流和竞争,优胜劣汰。

3. 经济模型

经济系统在某种意义上类似于生物生态系统。在商品市场和理想市场中,进化决定于经济实体的决策。选择机制是市场奖励机制。进化是快速的,企业与消费者之间、企业之间主要是一种互相依赖的合作关系。决策者为了追求长远利益,可以采取各种有效的方法,甚至可以暂时做赔本买卖。

14.6.6 基于对策论的协商

在多主体系统中,协商的含意有多种理解。一种认为子问题和资源的指派是协商。另一种则认为主体之间一对一直接协商。所有协商活动的目的是在一组独立工作的主体间构建协作,主体也有自己的目标。协商协议提供可能的协商形式的基本规则、协商过程和通信基础。协商策略取决于具体的主体。尽管主体开发者可以提供不同程度的协商能力,但是一定要保证协议与策略相匹配,即选择的策略要在可用的协议中能执行。

从单个主体看,协商的目的是改善自己的状态,在不影响自己情况下支持其它主体,或者对其它主体请求帮助。主体必须进行折衷,维护整个系统的能力。在这种意义上,协商交互的形式可以分成倍增长类:

(1) 对称协作。协商产生的结果,对每个主体都比它们原来达到的好。其它主体对主体本身的影响是积极的。

(2) 对称折衷。主体宁可自己独立达到它们的目标。协商意味着参加者之间的折衷,降低效果。但是不能忽略其它主体的存在,只能采取折衷,让参加者都能接受协商的结果。

(3) 非对称协作/折衷。即对协商的一个主体协作的影响是积极的,而对另一个主体必

须进行折衷。

(4) 冲突。由于主体的目标彼此冲突，不能达到可接受的解。在得到结果前协商必须终止。

14.6.7 基于意图的协商

Grosz 将主体的信念、愿望、意图理论应用到协商中[80]。该方法中不使用子规划，而是使用意图进行协商，减少通信量。BDI 理论认为，导致主体理性行为的既不是愿望，也不是规划，而是信念与愿望结合产生的意图，处于愿望与规划之间的层次。实现意图的子规划是由该意图产生。一个意图可能对应几个子规划。主体进行协商时没有必要交换各个子规划，只要交换意图。但是，Grosz 方法中假定主体是完全合作关系。

Zlotkin 的工作极大地改善了两个主体的静态协商理论。但是，这对开放的多主体系统不适用。清华大学王学军等用对策论重新定义意图，然后主体通过交换子意图进行协商[302]。

14.7 移动主体

随着 Internet 应用的逐步深入，特别是信息搜索、分布式计算以及电子商务的蓬勃发展，人们越来越希望在整个 Internet 范围内获得最佳的服务，渴望将整个网络虚拟成为一个整体，使软件 Agent 能够在整个网络中自由移动，移动 Agent 的概念随即孕育而生。

20 世纪 90 年代初，General Magic 公司在推出其商业系统 Telescript 时第一次提出了移动 Agent 的概念，即一个能在异构网络环境中自主地从一台主机迁移到另一台主机，并可与其它 Agent 或资源交互的软件实体。移动 Agent 是一类特殊的软件 Agent，它除了具有软件 Agent 的基本特性——自治性、响应性、主动性和推理性外，还具有移动性，即它可以在网络上从一台主机自主地移动到另一台主机，代表用户完成指定的任务。由于移动 Agent 可以在异构的软、硬件网络环境中自由移动，因此这种新的计算模式能有效地降低分布式计算中的网络负载、提高通信效率、动态适应变化了的网络环境，并具有很好的安全性和容错能力。

移动 Agent 可以看成是软件 Agent 技术与分布式计算技术相结合的产物，它与传统网络计算模式有着本质上的区别。移动 Agent 不同于远程过程调用 (RPC)，这是因为移动 Agent 能够不断地从网络中的一个节点移动到另一个节点，而且这种移动是可以根据自身需要进行选择的。移动 Agent 也不同于一般的进程迁移，因为一般来说进程迁移系统不允许进程自己选择什么时候迁移以及迁移到哪里，而移动 Agent 却可以在任意时刻进行移动，并且可以移动到它想去的任何地方。移动 Agent 更不同于 Java 语言中的 Applet，因为 Applet 只能从服务器向客户机做单方向的移动，而移动 Agent 却可以在客户机和服务器之间进行双向移动。

虽然目前不同移动 Agent 系统的体系结构各不相同，但几乎所有的移动 Agent 系统都包含移动 Agent (简称 MA) 和移动 Agent 服务设施 (简称 MAE) 两个部分。MAE 负责为 MA 建立安全、正确的运行环境，为 MA 提供最基本的服务 (包括创建、传输、执行)，实施针对具体 MA 的约束机制、容错策略、安全控制和通信机制等。MA 的移动性和问题求解能力很大程度上取取决于 MAE 所提供的服务，一般来讲，MAE 至少应包括以下基本服务：

- 事务服务 实现移动 Agent 的创建、移动、持久化和执行环境分配；
- 事件服务 包含 Agent 传输协议和 Agent 通信协议，实现移动 Agent 间的事件传递；
- 目录服务 提供移动 Agent 的定位信息，形成路由选择；
- 安全服务 提供安全的执行环境；

- 应用服务 提供面向特定任务的服务接口。

通常情况下，一个 MAE 只位于网络中的一台主机上，但如果主机间是以高速网络进行互联的话，一个 MAE 也可以跨越多台主机而不影响整个系统的运行效率。MAE 利用 Agent 传输协议 (Agent Transfer Protocol, ATP) 实现 MA 在主机间的移动，并为其分配执行环境和服务接口。MA 在 MAE 中执行，通过 Agent 通信语言 (Agent Communication Language, ACL) 相互通信并访问 MAE 提供的各种服务。

在移动 Agent 系统的体系结构中，MA 可以细分为用户 Agent (User Agent, UA) 和服务 Agent (Server Agent, SA)。UA 可以从一个 MAE 移动到另一个 MAE，它在 MAE 中执行，并通过 ACL 与其它 MA 通信或访问 MAE 提供的服务。UA 的主要作用是完成用户委托的任务，它需要实现移动语义、安全控制、与外界的通信等功能。SA 不具有移动能力，其主要功能是向本地的 MA 或来访的 MA 提供服务，一个 MAE 上通常驻有多个 SA，分别提供不同的服务。由于 SA 是不能移动的，并且只能由它所在 MAE 的管理员启动和管理，这就保证了 SA 不会是“恶意的”。UA 不能直接访问系统资源，只能通过 SA 提供的接口访问受控的资源，从而避免恶意 Agent 对主机的攻击，这是移动 Agent 系统经常采用的安全策略。

移动 Agent 是一个全新的概念，虽然目前还没有统一的定义，但它至少具有如下一些基本特征：

- 身份唯一性
移动 Agent 必须具有特定的身份，能够代表用户的意愿。
- 移动自主性
移动 Agent 必须可以自主地从一个节点移动到另一个节点，这是移动 Agent 最基本的特征，也是它区别与其他 Agent 的标志。
- 运行连续性
移动 Agent 必须能够在不同的地址空间中连续运行，即保持运行的连续性。具体来说就是当移动 Agent 转移到另一节点上运行时，其状态必须是在上一节点挂起时那一刻的状态。

移动 Agent 目前已经从理论探索进入到实用阶段，涌现出了一系列较为成熟的开发平台和执行环境。理论上移动 Agent 可以用任何语言编写（如 C/C++、Java、Perl、Tcl 和 Python 等），并可在任何机器上运行，但考虑到移动 Agent 本身需要对不同的软硬件环境进行支持，所以最好还是选择在一个解释性的、独立于具体语言的平台上开发移动 Agent。Java 是目前开发移动 Agent 的一门理想语言，因为经过编译后的 Java 二进制代码可以在任何具有 Java 解释器的系统上运行，具有很好的跨平台特性。

移动 Agent 技术虽然已经研究了很多年，但直到 1996 年才出现了真正实用的移动 Agent 系统，目前使用的移动 Agent 系统大致可以分为三类：一类是基于传统解释语言的，一类是基于 Java 语言的，另一类则是基于 CORBA 平台的。下面介绍几个典型的移动 Agent 系统，它们代表了当今移动 Agent 技术的基本方向和潮流：

1. General Magic 公司的 Odysseys

作为移动 Agent 系统专用语言的最早尝试，General Magic 公司开发的 Telescript 曾经在过去的几年里被广泛采用。Telescript 是一种面向对象的解释性语言，用它编写的移动 Agent 在通信时可以采用两种方式：若在同一场所运行，Agent 间可以相互调用彼此的方法；若在不同场所运行，Agent 间需要建立连接，互相传递可计算的移动对象。Telescript 在开始出现时还是一个比较成功的移动 Agent 开发平台，其安全性和健壮性都比较好，执行效率也很高，Telescript 中的三个基本概念 (agent、place 和 go) 对移动 Agent 做了一个很精辟的阐述：代理自主移动 (agent go place)。

随着 Java 的迅速崛起及其跨平台特性的逐步完善，Telescript 的优势慢慢消失，General

Magic 公司开始改变其策略, 开发了一个完全用 Java 实现的移动 Agent 系统 Odyssey, 它能够支持 Java RMI, Microsoft DCOM, 以及 CORBA IIOP。Odyssey 继承了 Telescript 中的许多特性, 是目前被广泛使用的一个移动 Agent 开发平台。

2. IBM 公司的 Aglet

Aglet 是最早基于 Java 的移动 Agent 开发平台之一, Aglet 的名字来源于 Agent 和 Applet, 可以简单地将其看成具有 Agent 行为的 Applet 对象。Aglet 以线程的形式产生于一台机器, 需要时可以随时暂停正在执行的工作, 并将整个 Aglet 分派到另一台机器上, 然后继续执行尚未完成的任务。从概念上讲, 一个 Aglet 就是一个移动 Java 对象, 它支持自动运行的思想, 可以从一个基于 Aglet 的主机移动到其它支持 Aglet 的主机上。

Aglet 构造了一个简单而全面的移动 Agent 编程框架, 为移动 Agent 之间的通信提供了动态而有效的交互机制, 同时还具备一整套详细而易用的安全机制, 这一切使得移动 Agent 的开发变得相对简单起来。

3. Recursion 公司的 Voyager

Voyager 可以看成是一个增强了的对象请求代理 (ORB), 同其它移动 Agent 系统相比, Voyager 与 Java 语言的结合更加紧密, 既可用于开发移动 Agent 系统, 也可用于创建传统的分布式系统。Voyager 是一个纯 Java 分布式计算平台, 可用来迅速生成高性能分布式应用程序, 是代表当前技术水平的一个优秀的移动 Agent 开发平台。

14.8 多主体环境 MAGE

多主体环境 MAGE 是一种面向主体的软件开发、集成和运行环境, 为用户提供一种面向主体的软件开发和系统集成模式, 包括面向主体的需求分析、系统设计、主体生成以及系统实现等多个阶段。它提供了多种软件重用模式, 可以方便地重用以前不同语言编写的主体或非主体软件; 它还提供了面向主体的软件开发模式, 以主体为最小粒度, 通过封装和自动化主体一般性质, 程序员可以通过特殊行为的添加方便地实现自己的应用; 这样, 通过构建新的软件以及重用旧的软件, 应用程序员可以方便地进各种应用集成。

14.8.1 MAGE 系统框架结构

MAGE 系统框架结构主要由三部分组成, 包括三个主要的工具需求分析和建模工具 AUMP、可视化主体开发环境 VASstudio 以及主体运行支持环境 Agent Supporting Environment。MAGE 系统的框架结构如图 14.24 所示。AUMP 支持面向主体的需求分析和设计阶段得到软件系统的模型, VASstudio 在 AUMP 得到的模型基础上支持面向主体的进一步设计和开发从而开发出软件系统所需的主体, 行为, 最后将这些主体放到主体支持环境 Agent Supporting Environment 中运行。

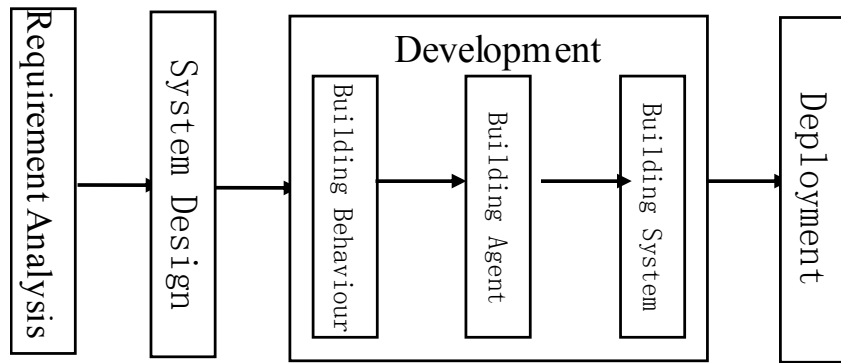


图 14.24 MAGE 系统框架结构

14.8.2 主体统一建模语言 AUML

主体统一建模语言(Agent Unified Modeling Language, 简称 AUML 或 Agent UML) 是一种面向主体的建模语言, 主要作用是帮助软件设计和开发人员对软件系统进行面向主体的描述和建模, 描述软件开发过程从需求分析直到实现和测试的全过程。AUML 将面向主体的软件开发方法和面向对象技术中的统一建模语言(Unified Modeling Language, 简称 UML) 结合起来。

AUMP 是运行于 Windows 操作系统上的多窗口应用程序。图形模型的设计和修改全部通过可视化的方法完成。

14.8.3 可视化主体开发环境 VASstudio

在基于主体的分布计算模型基础上, 将公共对象请求代理体系结构 CORBA 和 Internet 技术结合起来, 建立公共主体请求代理体系结构 CARBA (Common Agent Request Broker Architecture)。如下图所示, CABAR 主要由四部分构成: 软总线主体请求代理 ARB(Agent Request Broker)、主体应用框架 AppFacilities、主体领域模式 AppPattern、主体服务 AgentServices。

CARBA 是以主体请求代理 ARB (Agent Request Broker) 为核心的分布式构件管理机制, 它定义了分布式主体通过 ARB 透明地发送请求和接收响应的机制。主体应用框架 AppFacilities 将从水平和垂直方向提供主体构件。主体领域模式 AppPattern 是按照应用领域的需求, 建造各种具体的、与领域有关的模式或模板。主体服务 AgentServices 提供各种所需的主体服务, 如主体生存周期、主体库、命名、访问等。CARBA 可以很好地实现以下的目的:

- (1) 在异构分布式计算环境下按照功能分解系统, 划分系统框架;
- (2) 按照需要集成各个功能部件, 灵活组成系统。

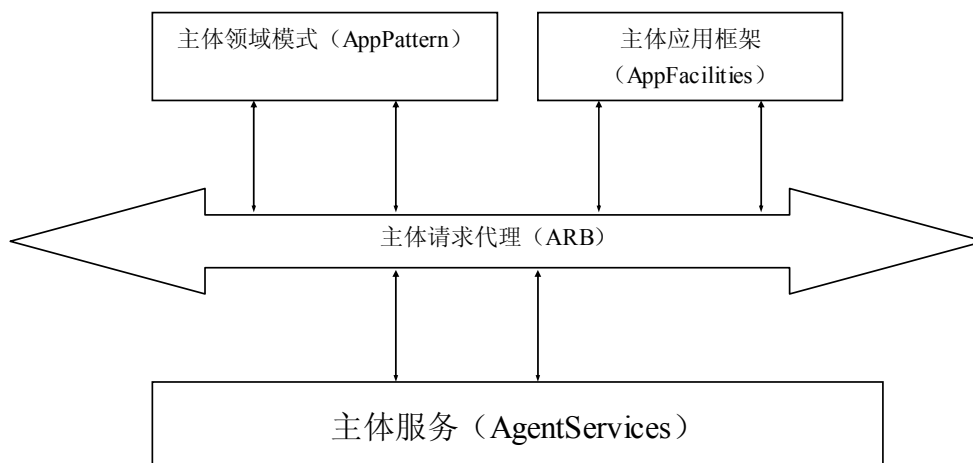


图 14.25 CARBA 的体系结构

基于上述思想，我们开发了一个可视化主体开发环境 VASudio，设计目标就是提供一个友好的集成环境来支持主体的设计和编程，不仅是系统编程环境，而且是面向主体的设计与编程环境，根据主体模型支持多种图形界面智能引导生成主体方式。同时为软件复用提供了一系列基本工具，如构件库管理工具、行为库、主体库、ADL 分析器以及本体编辑器等等。如图 14.26 所示。

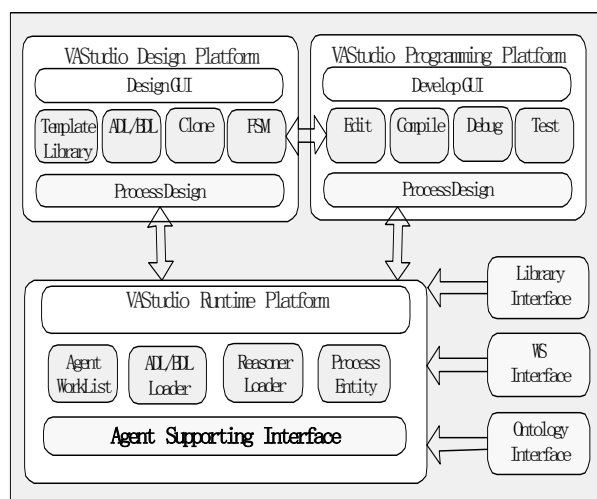


图 14.26 VASudio 体系结构

14.8.4 MAGE 运行平台

MAGE 平台框架遵循 FIPA 主体管理规范，是 FIPA 规范的一个标准参考实现。它提供主体创建、注册、定位、通信、移动和退出等服务，其体系结构如图 14.27 所示。

- 主体管理系统 Agent Management System (AMS)：是主体平台必须的组成部分。AMS 对主体平台的访问和使用提供管理和控制。一个主体平台上只能有一个 AMS。AMS 维持主体标识符的目录，包含了注册在主体平台上的主体的传输地址。AMS 给其他主体提供白页 (White-Page) 服务。每个主体都要向 AMS 注册，以得到一个有效的 AID。
- 目录服务器 Directory Facilitator (DF)：是主体平台必须的组成部分。DF 给其他主体提供黄页服务。主体可以在 DF 上注册它们的服务，并且可以在 DF 上查询其他主体提供了什么样的服务。一个主体平台可以有多个 DF。

- 消息传输系统 Message Transport Service (MTS)：为两个不同主体平台之间的通信提供传输服务。
- 主体 Agent：是主体平台上基本的元素。它是一个统一的、完整的执行模块，可以提供一项或多项服务，并且可以访问外部资源、用户界面和通信设施。每个主体都有主体标识符 (AID, Agent Identifier) 来唯一地标识。
- 主体平台 Agent Platform (AP)：为主体提供物理基础设施。AP 包括机器，操作系统，主体支撑软件，主体管理组件 (DF, AMS, MTS) 和主体。
- 软件 Software：指所有非主体、可执行的软件集合，通过主体可以进行访问。主体可以访问 Software，例如，添加新的服务、获得新的服务协议、获得新的安全协议/算法、获得新的协商协议、访问支撑工具等。
- 主体库和功能构件：用来组装生成主体的模板库。

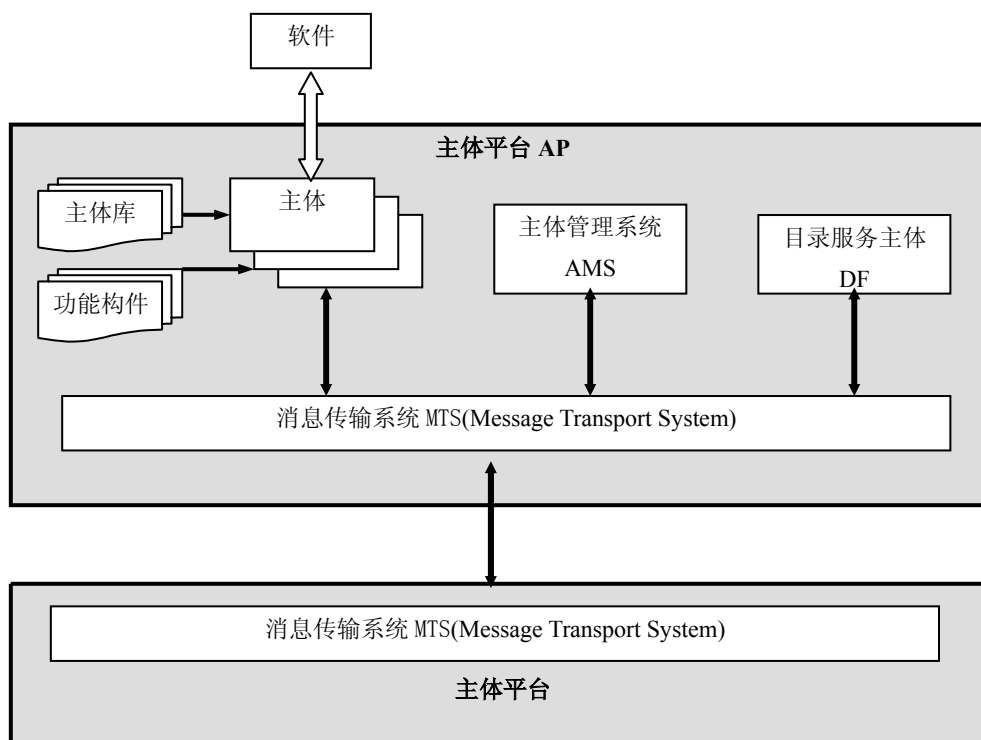


图 14.27 MAGE 主体平台的体系结构

习 题

1. 简述分布式问题求解系统的求解过程。
2. 简述智能主体的概念，以及主体的基本结构。
3. 什么是慎思主体？什么是反应主体？比较二者的区别。
4. 主体的 BDI 模型是什么，给出它的解释器算法。
5. 简述主体间是如何通信的。试比较主体通信语言 KQML 与 FIPA 的 ACL 语言的异同之处。
6. 分布式问题求解中合同网协议的原理是什么？
7. 如何实现多主体协同工作？
8. 解释移动主体的含义和基本特征，移动主体技术与传统网络计算模式有何区别。

9. 查阅资料，学习几个典型的移动主体系统。
10. 多主体环境 MAGE 由哪几部分组成？并给出他们的主要功能。
11. 主体统一建模语言 AUMML 在哪些方面扩展了统一建模语言 UML 的功能？
12. 请画出可视化主体开发环境 VASstudio 的框图？为什么他能有效地实现软件复用？

第十五章 进化计算

15.1 概述

进化计算(evolutionary computation)是研究利用自然进化和适应思想的计算系统[Yao 2006]。达尔文进化论是一种稳健的搜索和优化机制，对计算机科学，特别是对人工智能的发展产生了很大的影响。大多数生物体是通过自然选择和有性生殖进行进化。自然选择决定了群体中哪些个体能够生存和繁殖，有性生殖保证了后代基因中的混合和重组。自然选择的原则是适应者生存，不适应者被淘汰。

自然进化的这些特征早在 20 世纪 60 年代就引起了美国 Michigan 大学的 Holland 的极大兴趣。在那期间，他和他的学生们在从事如何建立机器学习的研究。Holland 注意到学习不仅可以通过单个生物体的适应实现，而且可以通过一个种群的许多代的进化适应发生。受达尔文进化论思想的影响，他逐渐认识到在机器学习中，为获得一个好的学习算法，仅靠单个策略的建立和改进是不够的，还要依赖于一个包含许多候选策略的群体的繁殖。考虑到他们的研究想法起源于遗传进化，Holland 就将这个研究领域取名为遗传算法(genetic algorithm)。一直到 1975 年 Holland 出版了那本颇有影响的专著“Adaptation in Natural and Artificial Systems” [Holland 1975]，遗传算法才逐渐为人所知。该书系统地论述了遗传算法的基本理论，为遗传算法的发展奠定了基础。同年，De Jong 的博士论文“An Analysis of the Behavior of a Class of Genetic Adaptive Systems”，把 Holland 的模式理论与自己的实验结合起来，得到一些很有意义的结论和方法，对以后遗传算法的应用和发展产生了很大的影响。

遗传算法思想来源于生物进化过程，它是基于进化过程中的信息遗传机制和优胜劣汰的自然选择原则的搜索算法(以字符串表示状态空间)。遗传算法用概率搜索过程在该状态空间中搜索，产生新的样本。遗传算法与自然进化的比较列于表 15.1。

表 15.1 遗传算法与自然进化的比较

自然界	遗传算法
染色体	字符串
基因	字符, 特征
等位基因 (allele)	特征值
染色体位置 (locus)	字符串位置
基因型 (genotype)	结构
表型 (phenotype)	参数集, 译码结构

人工遗传系统的关键是适应过程，它不是通过单个结构的递增变化，而是通过一个群体结构，通过遗传操作，诸如杂交、变异，产生新的结构。在群体中每个结构都有适应值，在竞争中用来决定哪些结构产生新的结构。

最简单的基于遗传算法的学习系统如图 15.1 所示，它由两个子系统构成：一个基于遗传算法的学习子系统，它将使结构产生适当的变化；另一个是执行子系统，它使系统行为得到改善。

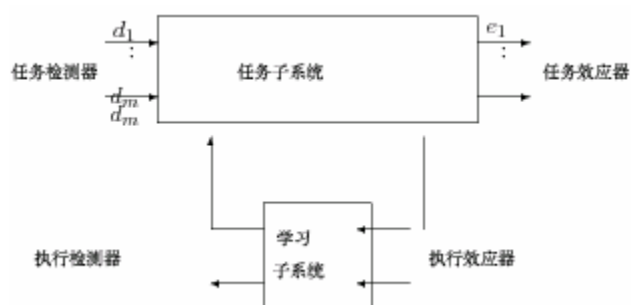


图 15.1 面向执行的学习系统

1988 年 Mayr 提出了新达尔文进化理论，其主要论点如下：

- (1) 个体是基本的选择目标；
- (2) 随机过程在进化中起重大作用，遗传变异大部分是偶然现象；
- (3) 基因型变异大部分是重组的产物，特别是突变；
- (4) 逐渐进化可能与表型不连续有关；
- (5) 不是所有表型变化都是自然选择的必然结果；
- (6) 进化是在适应中变化的，形式多样，不仅是基因的变化；
- (7) 选择是概率型的，而不是决定型的。

目前，进化计算主要包括遗传算法、进化策略、进化程序设计。下面分别进行介绍。

15.2 进化系统理论的形式模型

进化在个体群体中起作用。瓦铤顿 (Waddington) 指出基因型和表型之间关系的重要性 [Waddington 1974]。群体禁止异构环境。但是“后成环境”是多维空间。表型是基因型和环境的产物。然后表型通过异构“选择环境”发生作用。注意，这种多维选择环境与后成环境空间是不同的。现在，适应性是表型空间和选择环境空间的产物。它经常被取作一维，表示多少子孙对下一代作出贡献。

基于这种想法，莫楞贝 (Muhlenbein) 和肯德曼 (Kindermann) 提出了一种称为进化系统理论的形式模型 [Muhlenbein 1989]。图 15.2 给出了这种模型的关系。基因定义基因型空间 GS，表现性定义表型空间 PS。

$$GS = \{g = (a_1, \dots, a_n), a_i \in A_i\} \quad (15.1)$$

$$PS = \{p = (p_1, \dots, p_m), p_i \in IR\} \quad (15.2)$$

其中，g 是基因型，p 是表型。基因 g_i 的可能值称为等位基因。在门德尔 (Mendel) 遗传学中，假设每个基因有有限数的等位基因。

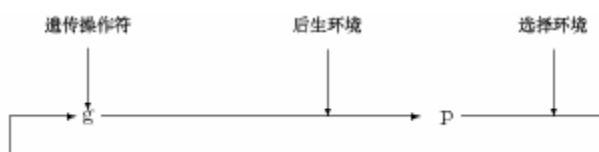


图 15.2 进化的主要过程

假设给定一组后生环境：

$$EP = \{EP_1, \dots, EP_k\}$$

和变换函数

$$\begin{aligned} f: GS \times EP &\rightarrow PS \\ p &= f(g, EP) \end{aligned} \quad (15.3)$$

这个变换函数给出了模型，说明表型的发展是通过基因与环境的交互作用。变换过程是高度非线性的。最终的表型特性具备一切特征，而对于原始的基因产品是一种简单的一对一关系。质量函数 q 给出了具体选择环境 ES_i 下表型的质量，其定义如下：

$$q(p, ES_i, t) \rightarrow IR^+ \quad (15.4)$$

质量定义适应度，用于达尔文选择。至今已有三种具体范例的通用模型，即

门德尔遗传学

遗传生态学

进化配子

在门德尔遗传学中，基因型被详细模型化，而表型和环境几乎被忽略。在遗传生态学中恰好相反。进化配子论是从社会生物学导出的模型。

首先让我们讨论门德尔遗传学的选择模型。为了简单起见，我们假设一个基因具有 n 等位基因 a_1, \dots, a_n 。二倍基因型以元组 (a_i, a_j) 为特征。我们定义 $p_{i,j}$ 为总群体中基因型 (a_i, a_j) 的频度。假设基因型与表型相等。质量函数给每个表现型赋值。

$$q(a_i, a_j)_i = q_{i,j}$$

其中 $q_{i,j}$ 可以被解释为出生率减去死亡率。

假设 $p'_{i,j}$ 是下一代表型 (a_i, a_j) 的频度。然后达尔文选择根据选择方程调整表型的分布：

$$p'_{ij} = p_{ij} \frac{q_{i,j}}{\bar{Q}} \quad (15.5)$$

$$\bar{Q} = \sum_{i,j} q_{i,j} p_{i,j} \quad (15.6)$$

其中 \bar{Q} 是群体的平均适应度。设 p_i 是群体中等位基因的频率。如果设

$$p_{i,j} = p_i p_j$$

那么，我们得到在 GS 中的一个选择方程为：

$$p'_i = p_i Q_i / \bar{Q} \quad (15.7)$$

$$Q_i = \sum_j q_{i,j} p_j \quad (15.8)$$

这个离散的选择方程可以用连续方程近似：

$$\frac{dp_i}{dt} = p_i (Q_i - \bar{Q}) / \bar{Q} \quad (15.9)$$

如果 $q_{i,j} = q_{j,i}$ ，那么

$$\frac{dp_i}{dt} = p_i (Q_i - \bar{Q}) \quad (15.10)$$

这个方程很容易被证明：

$$\frac{d\bar{Q}}{dt} = 2(E(Q^2) - \bar{Q}^2) = 2\text{Var}(Q) \geq 0 \quad (15.11)$$

这个结果称作菲希尔(Fisher)基本定理。它说明平均适应度随适应度的差别呈正比例增加。实际上，全部可能的基因型仅有一部分实现。这就是遗传操纵子探索基因型空间的任务，其个体数目相当小。这些操纵子是群体遗传变异性的来源。最重要的操纵子是突变和重组。

在生态遗传学中，人们假定显性是通过许多基因位点的作用而继承。在每个位点，两个等位基因分离，其中的显性值一个增加，另一个减少。

个体表型值的表达式是：

$$P_{i,j} = \mu + G_i + E_{i,j} + (GE)_{i,j}$$

其中 μ 是显性的总平均， G_i 是 i th 个表型的作用， $E_{i,j}$ 是基因型 i 的第 j 个个体环境的作用。

GE 是基因型与环境的交互作用。

组成的复杂关系阻止群体对某些“最佳表型”的直接存取。这种关系约束选择。重要约束是没有遗传变异，基因的相互关系和从基因型到表型的非线性变换。

在进化配子论中，每个个体能适应 N 种对策。设 p_i 是相继代中个体适应对策 i 的频率， $E_{i,j}$ 是个体适应对策 i 抵制对于适应对策 j 的赢利。个体适应对策 i 的品质为

$$Q_i = \sum_j p_j E(i, j)$$

群体平均适应度是

$$\bar{Q} = \sum_i p_i Q_i$$

假定个体无性生殖在数量上与它们的适应度成正比。这能够用熟知的选择方程：

$$p'_i = p_i \frac{Q_i}{\bar{Q}} \quad (15.12)$$

在进化配子论中寻找动力学吸引子。这些吸引子称作进化稳定对策(ESS)。对于 $p_i, p_j \neq 0$ ，ESS 的特征是 $Q_i = Q_j$ 。

15.3 达尔文进化算法

根据定量遗传学，达尔文进化算法采用简单的突变/选择动力学。达尔文算法的一般形式可以描述如下：

$$(\mu / \rho, \lambda) \quad (\mu / \rho + \lambda) \quad (15.13)$$

其中，

μ 是一代的双亲数目， λ 为子孙数目。整数 ρ 称作“混杂”数。如果两个双亲混合他们的基因，则 $\rho = 2$ 。仅 μ 是最好的个体才允许产生子孙。逗号表示双亲们没有选择，加号表示双亲有选择。

算法的重要部分是突变的范围不固定，而是继承。它将通过进化过程自己适应。达尔文进化算法如下：

算法 15.1 达尔文进化算法。

- (1) 建立原始种体。
- (2) 通过突变建立子孙。

$$\begin{aligned} s'_1 &= sg_1 \\ x'_1 &= x + s'_1 Z_1 \\ &\vdots \\ s'_\lambda &= sg_\lambda \end{aligned}$$

$$x'_\lambda = x + s'_\lambda Z_\lambda$$

- (3) 选择：

$$Q(x) = \max_{1 \leq i \leq \lambda} \{Q(x'_i)\}$$

- (4) 返回到步骤(1)。

在达尔文算法中，随机向量 Z_i 一般有分布的分量， $g_i s$ 是从分布数的规范对数得到的。所以算法在近邻的双亲建立 λ 子孙。通过进化继承和适应近邻的性质。模型可以被扩展到 $2n$ 个基因，而个体突变的范围被控制在 n -维空间。

15.4 分类器系统

Holland 和他的同事提出了一种分类器系统的认知模型，其中的规则不是规则集，而是遗传算法操纵的内部实体。图 15.3 给出了分类器系统的一般结构，从分类器系统看学习，它由三层动作构成，即执行子系统、信用赋值子系统和发现子系统。

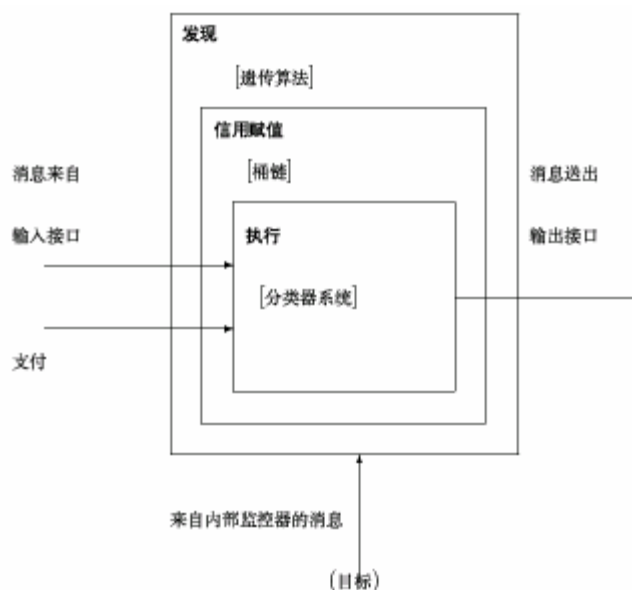


图 15.3 分类器系统的一般结构

执行子系统处在最低层，直接与环境进行交互。它与专家系统相同，由产生式规则构成。但是，它们是消息传送，高度平行。这类规则称作分类器。

分类器系统中的学习，要求环境提供反馈，确认所希望的状态是否达到。系统将评价这些规则的有效性，这些活动常常称作信用赋值。有些特定算法专门用来实现信用赋值，例如，桶链算法。

最后一层是发现子系统，该系统必须产生新的规则，取代当前用处不大的规则。通过系统累积的经验产生规则。系统根据适应值，使用遗传算法选择、重组和取代规则。

分类器系统是平行执行、消息传递和基于规则的系统。在简单的方案中，消息采用规定的字母，全部为固定长度。全部规则采用条件/动作形式。每个条件规定必须满足的信息，每个动作规定当条件满足时所发送的消息。为了方便，假设消息采用长度为 1 的二进制字符串记录，字符采用子集 {1, 0, #}。字符 # 表示无所谓，既可为 1，也可为 0。例如，字符串 11...1# 规定两个消息构成的子集，即 {11...11, 11...10}。更一般化形式为：

$$\langle s_1, s_2, \dots, s_j, \dots, s_l \rangle. \quad s_j \in \{1, 0, \#\}$$

它是由 1 个字符规定的子集，设

$$\langle m_1, m_2, \dots, m_j, \dots, m_l \rangle. \quad m_j \in \{1, 0\}$$

是 l 位消息，消息属于下列子集：

- (1) If $s_j = 1$ or $s_j = 0$, then $m_j = s_j$
- (2) If $s_j = \#$, then m_j can be either 1 or 0.

满足要求的全部消息构成子集，即每个子集是在消息空间的一个超平面。分类器系统是

由一组分类器 $\{C_1, C_2, \dots, C_N\}$ 、一个消息表、输入接口、输出接口构成。图 15.4 给出了分类器系统的基本结构。每部分的主要功能如下：

- (1) 输入接口将当前环境状态翻译成标准消息。
- (2) 分类器根据规则，规定系统处理消息的过程。
- (3) 消息表包含当前全部消息。
- (4) 输出接口将结果消息翻译成效应器动作，修改环境状态。

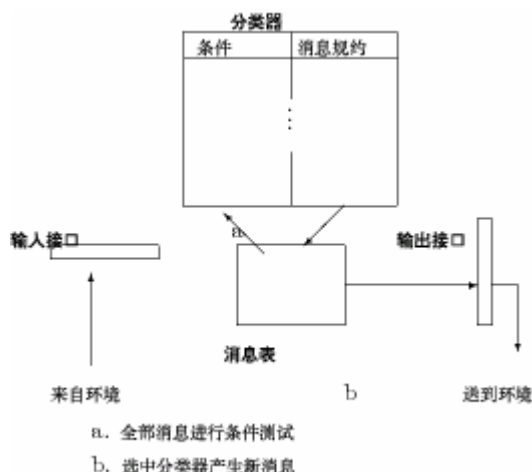


图 15.4 分类器系统的基本结构

分类器基本执行循环如下：

算法 15.2 分类器基本算法。

- (1) 将输入接口全部消息放入消息表。
- (2) 将消息表中的全部消息与全部分类器所有条件比较，记录所有匹配。
- (3) 满足分类器条件部分的每组匹配，将其动作部分所规定的消息送到新的消息表。
- (4) 用新的消息表取代消息表中的全部消息。
- (5) 将消息表中的消息翻译成输出接口的要求，产生系统当前的输出。
- (6) 返回到步骤(1)。

为了说明分类器的交互作用和基于规则学习的影响，这里介绍一个简单的视觉分类器系统，如图 15.5 所示。对视野内的每个对象输入接口产生消息。一组检测器产生消息，插入到各种性能。检测器和它们产生的值将按实际需要定义。系统有三类效应器，决定在环境中的动作。一类效应器控制视觉向量，该向量表示视野中心的定位。视觉向量可以按时间步递增式地旋转(V-LEFT 或 V-RIGHT，每次 15 度)。系统也有运动向量，表示它的运动方向，经常视觉方向是独立的。第二个向量控制运动向量的旋转(M-LEFT 或 M-RIGHT)。第二效应器也可以将运动向量与视觉向量对齐，或者处于相反方向(ALIGN 和 OPPOSE)。第三个效应器在方向设置运动速率(FAST, CRUISE, SLOW, STOP)。分类器处理检测器产生的信息，提供效应器命令序列，使系统到达目标。检测器规定输入接口消息最右边的 6 位(见图 15.5)。性质检测器规定的值如下所示：

$$d_1 = \begin{cases} 1, & \text{如果移动对象} \\ 0, & \text{其它} \end{cases}$$

$$(d_2, d_3) = \begin{cases} (0, 0), & \text{如果对象在视野的中间} \\ (1, 0), & \text{如果对象在中心的左边} \\ (0, 1), & \text{如果对象在中心的右边} \end{cases}$$

$$d_4 = \begin{cases} 1, & \text{如果系统是对象的近邻} \\ 0, & \text{其它} \end{cases}$$

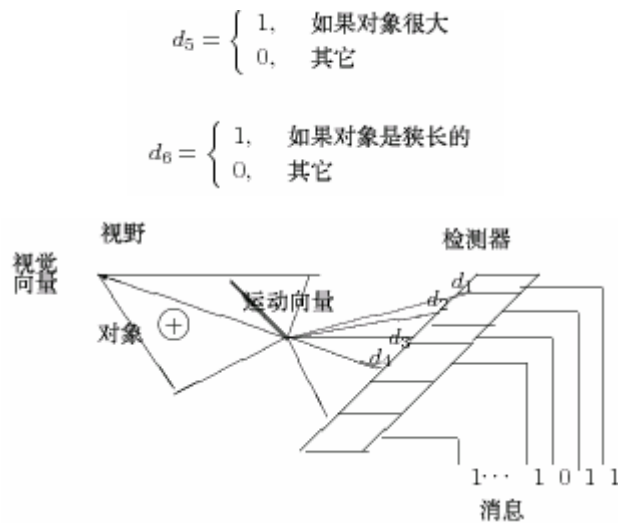


图 15.5 简单的视觉分类器系统}

考虑一个刺激——反应分类器，具有如下规则：

IF = 如果有“捕食(pre)” (small, moving, nonstriped object),
处于视野中间(centered), 非邻近 (nonadjacent),
THEN 迅速移向对象 (ALIGN), (FAST).

为了实现这条规则，作为分类器以检测值作为条件，其它采用#’s。两位(0, 0) 标签表示输入接口产生的消息。按照上述定义，分类器有下列条件：

00#####000001,

其中最左边两位是标签，# 位表示无关，最右边 6 位是规定的检测器值。当这个条件满足时，分类器送出消息如下：

0100000000000000,

其中前缀 01 表示该消息不是来自输入接口。这个消息可以直接用来设置输出接口的效应器条件，整个规则如下：

00#####000001 / 0100000000000000, ALIGN, FAST.

在一种警报系统中，下面的规则可以用来决定系统是否处于警报状态：

IF 在视野内有一个移动对象，

THEN 设置警报定时器和发出一个警报消息.

IF 警报定时器不是零，

THEN 发出一个警报消息.

IF 在视野内没有移动对象和警报定时器不为零，timer is not zero,

THEN 取消警报定时器.

为了将这些规则变成分类器，可以采用两种效应器，即 SET ALERT 和 DECREMENT ALERT，以及一个检测器

$$d_9 = \begin{cases} 1, & \text{if 警报定时器不为零} \\ 0, & \text{其它} \end{cases}$$

分类器实现三种规则，形式如下：

00#####1 / 010000000000011, SET ALERT

00#####1 / 010000000000011

00#####1 / DECREMENT ALERT.

分类器系统可以表示网络。最直接的方法是对网络中的每个点(向量)采用一个分类，图

15.6 给出了一种网络图。当在视野内出现一个 MOVING 对象，那就得到 ALERT 结点标识。当三个结点 ALERT, SMALL, 和 NOT STRIPED 有标识时, TARGET 结点才有标识。为了把这种网络变换成一组分类器, 可以对每个结点一个辨识标签。图中采用 5 位前缀作为标签。

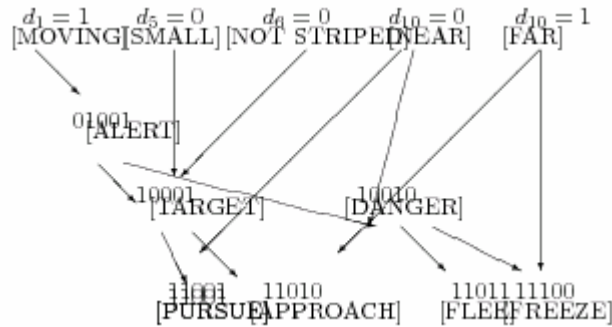


图 15.6 网络图

从图 15.6 可知, 假设 MOVING 结点被检测器 d_1 标识, 则 MOVING 和 ALERT 之间的箭头可以通过如下分类器实现:

00#####1 / 01001#####,

而从 SMALL, NOT STRIPED, and ALERT 到 TARGET 的箭头可以用单个分类器实现:

00#####00####, 01001##### /

10001#####.

网络中其它部分可以用类似方法实现。

15.5 桶链算法

分类器系统是高度并行的基于规则的学习系统, 它能连续地建造和改善它所在环境的模型, 这样的环境是在已有的经验的基础上建立的。而分类器系统的基本单元由消息和分类器(规则)构成, 形式上很简单。但这两个单元和缺省的分类器的组合可产生任意复杂的知识结构。

分类器系统使用两个学习机制,

(1) 桶链(bucket brigade) 算法。基于对系统的贡献, 对现有规则分配一个信用值。

(2) 规则发现算法。这包括遗传算法, 该算法可产生新规则, 用于改善系统的知识库。

因为强度也由分类器系统的执行部分所使用, 以便选择规则来控制系统的行为, 由规则发现系统来控制长期, 由桶链算法来分配它的强度。

根据分类器在系统达到目标方面整个效用性, 桶链算法赋给它信用值。通过修改执行周期, 算法工作引入分类器竞争。在工作周期里, 每个分类器扫描全局消息表中的所有消息。满足条件的每个消息产生新的消息。修改这个过程, 满足的分类器必须通过竞争从消息表中得到消息。每个分类器根据强度进行投标, 只有投标高的分类器才能从表中得到消息。投标规模不仅取决于分类器的强度, 也与它的特性有关。具体说, 分类器投标与强度和特性的乘积成正比。当分类器 C 的条件部分被满足, 投标的形式表达式为:

$$\text{Bid}(C, t) = cR(C)S(C, t) \quad (15.14)$$

其中 $R(C)$ 是特性, 等于 C 中条件部分的非# 的数目除以长度。 $S(C, t)$ 是 C 在时间 t 的强度, c 是一个小于 1 的常数, 例如 $1/8$ 或 $1/16$ 。

获胜的投标者把消息放到消息表中, 同时它们的强度减少投标量。例如对分类器 C :

$$S(C, t+1) = S(C, t) - \text{Bid}(C, t) \quad (15.15)$$

而被获胜者送去消息匹配的分类器 $\{C'\}$ 将提高它们的强度，其值为投标的量：

$$S(C', t+1) = S(C', t) + a \text{Bid}(C, t) \quad (15.16)$$

其中， $a = 1/(\{C'\} \text{成员数})$ 。

图 15.7 给出了投标的两个目的。支持服务，投标随消息流移动，增加对那些消息的用户的支持；同样投标用来修改供给者的强度，与消息流相反，进行反馈。

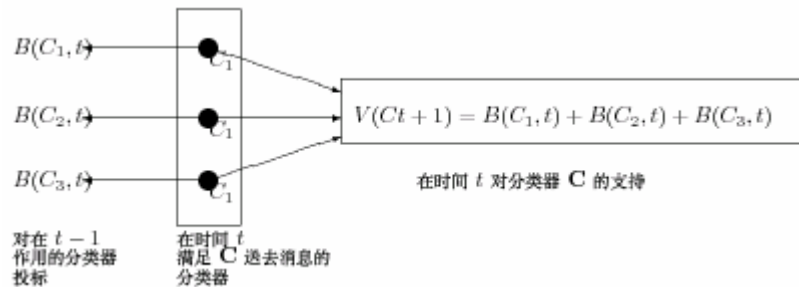


图 15.7 投标改变分类器的强度

假如系统的学习和稳定性可获得，并保持在可接受的水平，则可以减少规则强度的分散。假如不能控制分散，由规则发现作用于系统则会使系统采用强度高的规则，而使有用的、强度低的规则遗失掉。改变投标竞争和消息产生机制，有些改变要求控制个体的“寄生”规则，而另一些改变要求能保证有效地使用规模有限的消息表。

15.6 遗传算法

Holland 抽象地分析了自然系统的自适应过程，并设计出含有自然系统的自适应特征的人工系统。他将自适应过程刻画为在变化的结构空间中的搜索过程，指出：①许多复杂结构可以用简单的位串形式编码表示；②可以通过一些简单的变换来逐步改进这些位串结构，使之向希望的方向转变。并以这些思想为基础设计了遗传算法来实现结构的进化过程。

遗传算法先将搜索结构编码为字符串形式，每个字符串结构被称为个体。然后对一组字符串结构(被称为一个群体)进行循环操作。每次循环被称作一代，包括一个保存字符串中较优结构的过程和一个有结构的、随机的字符串间的信息交换过程。类似于自然进化，遗传算法通过作用于染色体上的基因寻找好的染色体来求解问题。与自然界相似，遗传算法对求解问题的本身一无所知，它所需要的仅是对算法所产生的每个染色体进行评价，并基于适应值来选择染色体，使适应性好的染色体有更多的繁殖机会。在遗传算法中，位字符串扮演染色体的作用，单个位扮演了基因的作用，随机产生一个体字符串的初始群体，每个个体给予一个数值评价，称为适应度，取消低适应度的个体，选择高适应度的个体参加操作。常用的遗传算子有复制、杂交、变异和反转。

与传统的优化算法相比，遗传算法主要有以下几个不同之处：

- (1) 遗传算法不是直接作用在参变量集上，而是利用参变量集的某种编码；
- (2) 遗传算法不是从单个点，而是在群体中从一个点开始搜索；
- (3) 遗传算法利用适应值信息，无需导数或其它辅助信息；
- (4) 遗传算法利用概率转移规则，而非确定性规则。

遗传算法利用简单的编码技术和繁殖机制来表现复杂的现象，从而解决非常困难的问题。特别是由于它不受搜索空间的限制性假设的约束，不要求诸如连续性、导数存在和单峰等假设，它能从离散的、多极值的、含有噪音的高维问题中以很大的概率找到全局最优解；其

次，由于它固有的并行性，遗传算法非常适用于大规模并行计算。遗传算法目前已经在优化、机器学习和并行处理等领域得到了越来越广泛的应用。

15.6.1 遗传算法的主要步骤

为了使用遗传算法来解决问题，准备工作分为以下四步：

- (1) 确定表示方案；
- (2) 确定适应值的度量；
- (3) 确定控制该算法的参数和变量；
- (4) 确定怎样指定结果及程序运行结束的标准。

在常规的遗传算法中，表示方案是把问题的搜索空间中的每个可能的点表示为固定长度字符串。表示方案的确定需要选择串长 l 和字母表规模 k ，二进制位串是遗传算法中常用的表示方法。适应值度量为群体中每个可能的确定长度的特征串指定一个适应值，它经常是问题本身所具有的。控制遗传算法的主要参数有群体规模 N 和算法执行的最大代数 M ，次要参数有复制概率 p_r 、杂交概率 p_c 、变异概率 p_m 等参数。有关停止执行的标准要视具体问题而定。一旦这些准备步骤完成后，就可以使用遗传算法。

算法 15.3 基本的遗传算法。

- (1) 随机产生一个由固定长度字符串组成的初始群体；
- (2) 对于字符串群体，迭代地执行下述步骤，直到选种标准被满足为止：
 - ① 计算群体中的每个个体字符串的适应值；
 - ② 应用下述三种操作(至少前两种)来产生新的群体：
 - 复制：把现有的个体字符串复制到新的群体中。
 - 杂交：通过遗传重组随机选择两个现有的子字符串，产生新的字符串。
 - 变异：将现有字符串中某一位的字符随机变异。
- (3) 把在后代中出现的最高适应值的个体字符串指定为遗传算法运行的结果。这一结果可以是问题的解(或近似解)。

基本的遗传算法流程图如图 15.8 所示，其中变量 GEN 是当前进化代数。

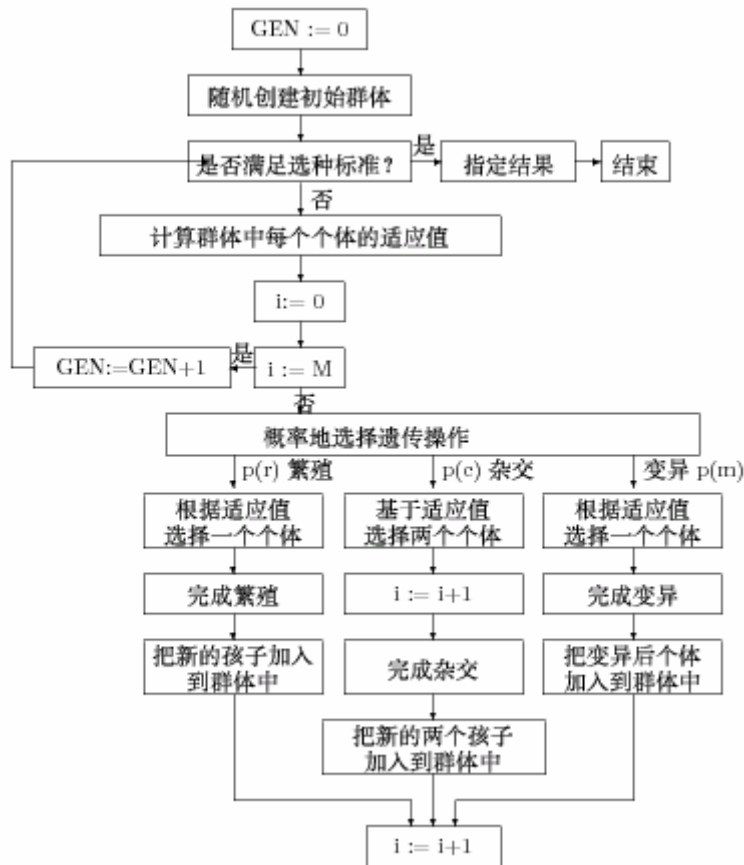


图 15.8 基本遗传算法的流程图

15.6.2 表示模式

在遗传算法中，可以考虑由三元字母表 $V_+ = \{0, 1, *\}$ 表示的模式。所谓模式就是一个相同的构形，它描述的是一个串的子集，这个集合中的串之间在某些位上相同。符号“*”代表不确定字母，即在一特定位置上与 0 或 1 均匹配。例如，考虑串长为 7 的模式 $H = *11*0**$ ，则串 $A = 0111000$ 是模式 H 的一个表式，串 A 在确定位置 2、3 和 5 上相匹配。

定义在串长为 l 的二进制串上共有 3^l 个模式。一般对于基数为 k 的字母表，共有 $(k+1)^l$ 个模式。一个模式 H 的阶就是出现在模式中确定位置的数目，记为 $o(H)$ 。在二进制串中，一个模式的阶就是所有 1 或 0 的数目。例如，模式 $011*1**$ 的阶为 4，可记为 $o(011*1**) = 4$ 。模式 $0*****$ 的阶为 1。

一个模式的定义长度是模式中第一个确定位置与最后一个确定位置之间的距离，记为 $\delta(H)$ 。例如，模式 $011*1**$ 的定义长度为 $\delta(H) = 4$ ，这是因为第一个确定位置为 1，最后一个确定位置为 5，它们之间的距离 $\delta(H) = 5 - 1 = 4$ 。

假设时刻 t ，一个特定的模式 H 有 M 个代表串包含在群体 $B(t)$ 中，记为 $M = M(H, t)$ 。

在复制阶段，每个串根据它的适应值进行复制，一个串 B_i 的复制概率为

$$p_i = \frac{f_i}{\sum_{j=1}^n f_j} \quad (15.17)$$

当采用非重叠的 n 个串的群体代替群体 $B(t)$ 后，我们期望在时间 $(t + 1)$ ，模式 H 在群体 $B(t+1)$ 中有 $M(H, t+1)$ 个代表串，这可由下面方程给出：

$$M(H, t+1) = M(H, t) \cdot n \cdot \frac{f(H)}{\sum_{j=1}^n f_j} \quad (15.18)$$

其中 $f(H)$ 是在时间 t 表示模式 H 的串的平均适应值。由于整个群体的平均适应值可记成

$$\bar{f} = \sum_{j=1}^n \frac{f_j}{n} \quad (15.19)$$

故模式的复制生长方程可表示为

$$M(H, t+1) = M(H, t) \frac{f(H)}{\bar{f}} \quad (15.20)$$

这表明，一个特定的模式按照其平均适应值与群体的平均适应值之间的比率生长。换句话说，那些适应值高于群体平均适应值的模式在下一代中将会有更多的代表串，而对于那些适应值在群体平均值以下的模式，它们的下一代中的字符串将会减少。假设模式 H 的适应值保持在高出群体平均适应值以上一个 $c\bar{f}$ ， c 为一常数，则模式的生长方程可变为

$$M(H, t+1) = M(H, t) \frac{\bar{f} + c\bar{f}}{\bar{f}} = (1+c) \cdot M(H, t) \quad (15.21)$$

从 $t = 0$ 开始，假设 c 是一个固定值，可以推得

$$m(H, t) = m(H, 0) \cdot (1+c)^t \quad (15.22)$$

(15.22) 式表明，在群体平均适应值以上的模式将会按指数增长的方式被复制，相反，在群体平均适应值以下的模式将会按指数衰减的方式被复制。

15.6.3 杂交操作

杂交操作是个有结构、随机的字符串间信息交换过程。假设群体 $B(t)$ 是模式库。历史信息以每个模式实例数目的形式存储于 $B(t)$ 中。杂交作用产生模式库中已有模式的新的实例，同时也产生新的模式。简单的杂交操作分为三步：

- (1) 从当前群体 $B(t)$ 中选择两个结构： $a = s_1 s_2 \cdots s_l$ ， $a' = s'_1 s'_2 \cdots s'_l$
- (2) 随机选择一个整数 $x \in \{1, 2, \cdots l-1\}$
- (3) 交换 a 和 a' 中位置 x 左边的元素，产生两个新的结构： $s_1 \cdots s_x s'_{x+1} \cdots s'_l$ 和 $s'_1 \cdots s'_x s_{x+1} \cdots s_l$

如果给每个字符串赋以强度 $S(C_j, T)$ ，那么，遗传算法如下所示：

算法 15.4 具有强度的遗传算法。

- (1) 在 $t = 0$ 时随机产生 M 字符串的群体 $B(t)$ 。计算群体 $B(t)$ 中字符串的平均强度 $v(t)$ ，给群体 $B(t)$ 中的每个字符串赋以规范值 $S(C_j, t)/v(t)$ 。

- (2) 对群体 $B(t)$ 中的每个字符串赋与一个概率值，其值与规范值成正比。然后，使用概率分布，从 $B(t)$ 中选择 n 对字符串， $n \ll M$ ，并且将它们复制。

- (3) 对每对复制字符串进行杂交操作，形成 $2n$ 个新的字符串。
- (4) 用步骤(3)中生成的 $2n$ 个新的字符串取代群体 $B(t)$ 中 $2n$ 个强度最低的字符串。
- (5) 时间 t 变为 $t + 1$ ，返回步骤(1)。

为了说明上述算法的操作，让我们看一个图 15.9 所示的简单例子。该例子中有如下 8 个样本：

```

C1  1 1 1 0 1 1 1 0 1 ... ← 1
C2  1 1 0 0 1 1 1 1 0 ... ← 0
C3  0 0 0 1 1 1 1 1 0 ... ← 2
C4  0 1 1 0 1 0 0 1 1 ... ← 0
C5  1 0 0 0 1 1 1 0 1 ... ← 2
C6  0 1 1 0 0 1 1 0 0 ... ← 2
C7  1 0 0 1 0 0 1 0 1 ... ← 0
C8  1 0 0 0 1 0 0 1 0 ... ← 1

```

其中，左箭头的右边值是群体 $B(t)$ 中每个字符串 C_i 的强度。样本 C_2, C_3, C_6 是模式 $*****1*0*...$ 的例示，称作 H_1 。样本 C_3, C_5, C_8 是模式 $*00*1*...$ 的例示，称作 H_2 。对每个模式计算平均强度：

$$v(H_1) = \frac{0+2+2}{3} = 1.33$$

$$v(H_2) = \frac{2+2+1}{3} = 1.67$$

这样，我们给每个字符串 C_i 赋与规范值 $S(C)/v(H)$ 。根据规范值赋给每个字符串一个概率。按照概率分布选择 3 对字符串进行杂交。字符串的位置是随机选择的，要进行杂交的字符串交换其该位置的左部。这种简单的操作产生微妙的影响。图 15.9 给出了对模式的整个影响。在图 15.9 中 C_2, C_3, C_6 是模式 $*****1*\#$ 的例示。该模式的平均强度如下：

$$v(*****1*\#) = \frac{0+2+2}{3} = 1.33$$

C_3, C_5, C_8 是模式 $*00*\#*****$ 的例示，平均强度如下：

$$v(*00*\#*****) = \frac{2+2+1}{3} = 1.67$$

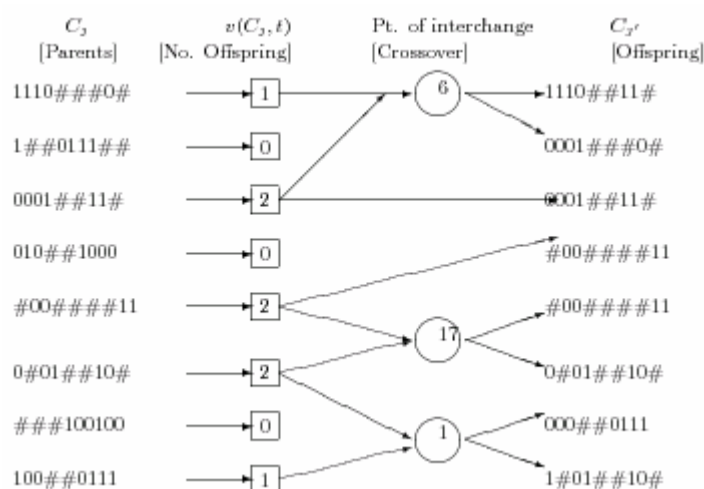


图 15.9 遗传算法操作

在遗传算法中只使用简单的杂交操作，那么 $B(t)$ 中所表示的每个模式 H 在群体中所占的

平均比例，经过一代后则从 $P(H, t)$ 变为

$$P(H, t+1) \geq (1-p_c \frac{\delta(H)}{l-1} (1-P(H, t))) \frac{f(H,t)}{\bar{f}(t)} P(H, t) \quad (15.23)$$

其中 p_c 是个体在一代中进行杂交操作的比率。可以推得，只要 $f(H, t) \geq [1 + \frac{\delta(H)}{l-1}] \bar{f}(t)$ ， H 的实例数就增加。

15.6.4 变异操作

对于群体 $B(t)$ 中的每个个体 $a = s_1 s_2 \cdots s_l$ ，简单的变异操作过程如下：

(1) 每个位置的字符变量都有一个变异概率，各位置互相独立。通过随机过程选择发生变异的位置 x_1, x_2, \cdots, x_l 。

(2) 产生一个新结构 $a' = s_1 \cdots s_{x_1-1} s'_{x_1} s_{x_1+1} \cdots s_{x_2-1} s'_{x_2} s_{x_2+1} \cdots s_l$ ，其中 s'_{x_1} 是从对应位置 x_1 的字符变量的值域中随机选择的一个取值。 $s'_{x_2}, \cdots, s'_{x_k}$ 可以同样得到。

如果每个位置的变异概率小于等于 p_m ，那么模式 H ，设阶为 $o(H)$ ，发生一次或多次变异的概率是

$$1 - (1-p_m)^{o(H)} \approx o(H) p_m, \quad (p_m \ll 1/l) \quad (15.24)$$

15.6.5 反转操作

简单反转操作的步骤如下：

- (1) 从当前群体中随机选择一个结构 $a = s_1 s_2 \cdots s_l$ ；
- (2) 从 $\{0, 1, \cdots, l+1\}$ 中随机选择两个数 i' 和 j' ，并定义 $i = \min\{i', j'\}$ ， $j = \max\{i', j'\}$ ；
- (3) 颠倒 a 中位置 i, j 之间的部分，产生新的结构

$$s_1 s_2 \cdots s_i s_{j-1} s_{j-2} \cdots s_{i+1} s_j \cdots s_l$$

反转操作能改变个体结构中字符变量的位置，把原来相距较远的字符变得较近。字符串的任意一种排列都能用适当的反转序列产生出来。反转操作对模式 H 的作用是随机改变 H 的长度，改变模式中有效字符变量间的联结性。

遗传操作除了有复制、杂交、变异和反转算子外，还有染色体内部复制(intrachromosomal duplication)、删除、易位(translocation)、分异(segregaon)等。

15.7 并行遗传算法

基于离散门德尔模型的遗传算法由五部分组成：

- (1) 对给定问题求解的染色体表示；
- (2) 求解的原始物种；
- (3) 起环境作用的品质函数；

- (4) 可以生成子孙的个体的选择过程;
- (5) 一种遗传操纵子, 如变异、重组;
- (6) 控制算法本身的参数。

在自然界中, 从个体生活的环境里进行选择是局部地完成的。从环境里抽象得到的自然选择方程会导致重复的危险。并行遗传算法成功地应用在许多组合问题中。每个物种的个体分别用处理机仿真, 所以并行地作用其他个体。基本算法如下:

- (1) 给定具有不同开始表型的 N 个个体;
- (2) 每个个体计算局部最大;
- (3) 选择——在“近邻”中选择配对;
- (4) 用重组和突变创建子孙;
- (5) 返回步骤 2。

并行算法与经典算法之间有两点差别。一是在近邻中选择配对是局部完成的。二是每个个体是爬山完成的。

所以并行遗传算法采用基因型学习和表型学习。更进一步, 个体是生活在环境中, 杂交和选择是在这种环境中完成, 例如 2-D 格网。个体的近邻与基因型和表型空间的格网没有连接。并行遗传算法主要是从遗传算法来的。在遗传算法中个体是被看成是“最佳行为”的选择。相反, 并行遗传算法中是从内部驱动。并行遗传算法比普通遗传算法导致更大的差异性。进化是系统内部自组织驱动的。

15.8 分类器系统 Boole

Wilson 于 1987 年开发了一个布尔问题的分类器系统 Boole[Wilson 1987]。一个布尔函数变量 L 是从长度为 L 的字符串到 $\{0, 1\}$ 的映射。学习函数意味获得对任何输入字符串能给出正确输出 0 或 1 的能力。

每个 Boole 分类器由一个分类名和一个动作组成。 L -位输入字符串, 分类名长度为 L 。动作是简单的一位 0 或 1, 表示函数两个可能值之一。如果分类名与当前输入字符串匹配, 系统作出决策。分类器群体 $[P]$ 通常根据某种随机规则, 初始时全部填写 0, 1, 或 #。动作也是同样填写。例如, 长度为 $L = 6$ 的分类器是:

1 0 # 0 # 1 / 1,

在 Boole 中, $[P]$ 包含 400 个分类器, 以一致分布初始化。初始强度为 100。

在执行周期时, 当输入字符串出现, 系统作出回答 0 或 1。周期分两步:

- (1) 与输入串匹配, 形成匹配集 $[M]$ 。
- (2) 对 $[M]$ 分类器的强度使用概率分布, 从 $[M]$ 中选择一个分类器, 即一个特定的分类器的选择概率等于它的强度除以 $[M]$ 中分类器强度的总和。被选的分类器的动作作为系统的决策。

增强部件在执行周期后, 根据从环境接受的赢利, 调整分类器的强度。Wilson 使用下面的算法。

算法 15.5 分类器强度调整算法[Wilson 1987]。

- (1) 将与所选动作相同的分类器形成子集 $[M]$, 称作动作集 $[A]$ 。将不在 $[M]$ 中的其它分类器放在集合 $\text{NOT}[A]$ 中。
- (2) 在 $[A]$ 中的全部分类器强度减少一个分数 e 。
- (3) 如果系统决策正确, 则将赢利量 R 分配给 $[A]$ 的强度;
- (4) 如果系统决策错误, 则将赢利量 R' (其中 $0 \leq R' \leq R$) 分配给 $[A]$ 的强度, 从 $[A]$

的强度减少一个分数 p 。至少 R' 和 p 中的一个为 0。

(5) 从 NOT[A] 中的强度减去一个分数 t 。

若步骤(3) 为正确决策, 对 [A] 的增强影响可用下式计算:

$$S'_A = S_A - eS_A + R \quad (15.25)$$

其中, S_A 为在赢利之前[A]的总强度, S'_A 是总的赢利。假设对于[A]所有时间 R 均相同, 可

以证明 S_A 将逐渐接近 R/e 。建议在相当稳定赢利条件下, S_A 将是典型赢利估值的函数。步

骤(4) 定义了 3 种不同的赢利机制: 赢利—惩罚, $p \neq 0$; 仅考虑赢利, p 和 R' 均为 0; 赢利—赢利, $R' \neq 0$ 。第一种反映环境, 表明系统的结果是错误的。第二种与环境响应, 在正确动作产生之前一般没有差别。第三种机制反映一般情况, 有一个得到最大赢利, 而其余动作得到相同赢利。步骤(2) 中强度减少的分数可以认为是[A]中接受赢利机会的每个分类器所付出的代价。[A]的总赢利是通过分布函数 D 分配, 最简单的情况给每个分类器等于共享 R 。

发现部件是基于 Holland 的遗传算法, 使用复制、遗传操作和删除。下面给出 Boole 的遗传算法。

算法 15.6 Boole 的遗传算法。

- (1) 根据[P]中分类器的强度, 通过概率选择第一个分类器 C_1 。
- (2) 根据概率 χ , 与步骤(1) 相同, 选择分类器 C_2 , 并对 C_2 和 C_1 进行杂交; 从结果中选择一个作为子孙, 另一个被抛弃。
- (3) 如果步骤 (2) 未完成, 则复制 C_1 形成子孙。
- (4) 对子孙应用变异操作, 以概率 μ 改变每个分类的等位基因。
- (5) 如果通过杂交生成子孙, 每个父母的强度减少三分之一, 子孙的初始强度等于父母减少的总和; 否则减少 C_1 的一半, 而子孙的初始强度等于减少的量。
- (6) 将子孙加到[P]中。
- (7) 根据[P]中分类器强度的概率分布, 删除最小的一个分类器。

每个试验, 即每个执行周期, 算法以概率 ρ 被调用。基本上与标准系统算法相同, 不同的是调用一次仅产生一个子孙, 而标准的算法通常产生许多子孙。

表 15.1 6-路复用器学习结果

例示号	“概念”								总强度
	(分类名)							(动作)	
56	0	1	#	0	#	#	/	0	7655
52	0	1	#	1	#	#	/	1	7541
48	0	0	0	#	#	#	/	0	7056
46	1	0	#	#	0	#	/	0	7095
45	0	0	1	#	#	#	/	1	6665
41	1	1	#	#	#	0	/	0	5964
39	1	1	#	#	#	1	/	1	6323
35	1	0	#	#	1	#	/	1	5145
7	#	1	#	1	#	1	/	1	1044
4	1	1	#	#	#	#	/	1	522
3	#	0	#	#	1	#	/	1	293
3	1	1	#	#	0	#	/	0	210
2	#	0	1	#	#	#	/	1	330
2	0	1	1	#	#	#	/	1	212
2	1	0	0	#	0	#	/	0	150
2	0	0	#	#	#	#	/	1	219
2	1	#	#	#	1	#	/	0	326
2	1	0	#	0	#	#	/	0	238
1	#	1	#	0	#	#	/	0	129
1	1	0	#	#	#	#	/	0	168
1	1	1	#	#	#	#	/	0	97
1	1	0	#	#	0	0	/	0	100
1	0	0	#	#	1	#	/	1	81
1	1	#	#	#	#	0	/	0	212
1	1	0	#	#	0	1	/	1	56
1	0	1	#	#	#	#	/	1	116

发现模块还有一个机理是创建，即当执行周期时[M]为空，就生成一个新的分类器。创建的新分类器名是当前输入字符串的复制，而插入 # 的概率等于当前[P]中 # 的百分数。为了得到[P]中的位置，可以采用步骤(7)的方法删除一个分类器。然后，[M] 重新计算，继续执行周期。

使用 Boole 学习 6-路复用器。每个整数 $k > 0$ ，二进制字符串的多路复用器的长度 $L = k + 2^k$ 。每个输入字符串有 k 个地址位 a_i 和 2^k 个数据位 d_i , 字符串表达式为

$$a_0a_1\cdots a_{k-1}d_0d_1\cdots d_{2^k-1}$$

函数值是通过数据位的 (0 或 1) 值给出。例如，下面是 6-路复用器输入字符串和正确输出值的函数， $L = 6$ ， $k = 2$ ：

```

0 0 0 1 0 1      0
1 1 0 0 0 1      1
1 0 1 1 0 1      0

```

因为 $k = 2$ ，第一个例子的前两个地址位是 00，告诉我们数据位输出值选为 0。第二个例子，地址位是 11，所以看数据位 3，因此值为 1。函数写成析取范式为

$$F_6 = a'_0a'_1d_0 + a'_0a_1d_1 + a_0a'_1d_2 + a_0a_1d_3$$

6—路复用器为：

```

0 0 0 # # # / 0

```

```

0 0 1 # # # / 1
0 1 # 0 # # / 0
0 1 # 1 # # / 1
1 0 # # 0 # / 0
1 0 # # 1 # / 1
1 1 # # # 0 / 0
1 1 # # # 1 / 1

```

这个集合称为[S6]。Boole 将发现特定分类器的多个成员。在试验中，采用赢利-惩罚策略，其参数为：e = 0.1, R = 1000, R' = 0, p = 0.8, G = 4.0, μ = 0.001, χ = 0.12, ρ = 1.0, t = 0.1。通过 15, 000 试验，结果在表 15.1 给出。顶上的 8 个分类器是[S6]的成员。

15.9 规则发现系统

在规则发现系统中，学习经常是首先评价系统现有的规则质量，然后进行修改。Grefenstette 研制了一种规则发现系统 RUDI。图 15.10 给出了 RUDI 的控制结构。问题求解级由简化的分类器系统组成。学习级是对知识结构群体进行遗传算法操作，每一个表示为一组规则表。知识结构的整个行为控制这些结构的复制。



图 15.10 RUDI 的控制结构

在 RUDI 中，信用赋值方法赢利共享规划(Profit-Sharing Plan, 简称 PSP) 和桶链算法(BBA) 对每个规则提供互补的效用信息。根据期望的外部奖励，PSP-强度对规则效用提供更精确的评估。当问题求解时它被用作冲突消解。与此相反，BBA-强度表示规则之间的动态相关性，规则点火依次会聚到相似水平。这种测度可以用作一组协作规则的聚类。

Grefenstette 提出一种强度修改方案称作赢利共享规划 PSP。在这种方案中问题求解划分成情节，按所接受的外部奖励区分。如果任何步情节在投标竞争中获胜，则认为该规则在该情节活动。在情节 t, PSP 修改每个活动规则 R_i 的强度 $S_i(t)$ 如下：

$$S_i(t+1) = S_i(t) - bS_i(t) + bp(t), \quad (15.26)$$

其中，p(t) 称作在情节结束时所获得的外部奖励，即当获得外部奖励，从每个活动规则搜集投标，每个活动规则给出一部分外部奖励。考虑 PSP 对给定规则 R_i 的影响，它按照方程(15.26) 得到：

$$S_i(t) = (1-b)^t S_i(0) + b \sum_{i=1}^t (1-b)^{t-i} p(i-1) \quad (15.27)$$

其中， t 的范围是在该情节规则 R_i 是活动的，即 $S_i(t)$ 基本上外部奖励的权值平均 $p(t)$ ， $(1-b)$ 作为指数衰减因子。如果 b 足够小，那么 $S(t)$ 具有 $p(t)$ 的平均值。如果外部奖励 $p(t)$ 是常数， p^* ，那么 S_i 收敛到一个平衡值 S_i^* ：

$$S_i^* = \lim_{t \rightarrow \infty} [(1-b)^t S_i(0) + b \sum_{i=1}^t (1-b)^{t-i} p^*] = p^* \quad (15.28)$$

在常数赢利下，它符合方程 (15.27)，PSP 将以下列速率减少误差 $E_i(t) = p^* - S_i(t)$ ：

$$\begin{aligned} \Delta E_i &= E_i(t+1) - E_i(t) \\ &= S_i(t) - S_i(t+1) \\ &= -b(p^* - S_i(t)) \\ &= -bE_i(t). \end{aligned} \quad (15.29)$$

强度每次改变，以因子 b 减少当前强度与平衡强度之差。例如，设 $p^* = 500$ ， $b = 0.1$ ，和 $S_i(t) = 100$ ，那么 $S_i(t+1) = 100 - 10 + 50 = 140$ 。注意，误差 $p^* - S_i$ 从 400 减到 360，即 10%。

我们看出，奖励相当是常数情况下，在 PSP 下每个规则强度很快收敛到一个平衡强度，可以预测情节结束时将接收的奖励水平。PSP 的一种可能的限制是它取决于这种前提，成功外部奖励区分的情节所对应的合适区间，在这个区间里进行信用赋值。情节的选择非常重要。

在桶链算法 BBA 中，是基于规则之间单独处理的，可以避免有关情节的假设。假设规则 R_i

在 τ 步点火，规则 R_j 在 $\tau + 1$ 点火，那么 BBA 按照下面公式修改规则 R_i 的强度 S_i ：

$$S_i(\tau+1) = S_i(\tau) - bS_i(\tau) + bS_j(\tau) \quad (15.30)$$

除情节指数 t 被步指数 τ 代替，外部奖励 $p(t)$ 被规则 R_j 的强度 S_j 取代外，这个公式与式 (15.26) 一样。第一个改变意味 BBA 在给定的情节修改规则强度多于一次。第二个改变导致 PSP 与 BBA 基本的不同。PSP 强度预测所期望的情节结束获得的外部奖励是在规则点火，BBA 的强度预测所期望的内部奖励是在规则的下一步。考虑两条规则 R_i 和 R_j ， R_i 每次点火紧跟着 R_j 点火，设 R_i 和 R_j 在每个情节最多一次，那么有：

$$\begin{aligned} S_i^* &= \lim_{t \rightarrow \infty} [(1-b)^t S_i(0) + b \sum_{i=1}^t (1-b)^{t-i} p^*] = p^* \\ S_i(t) &= (1-b)^t S_i(0) + \sum_{i=1}^t b(1-b)^{t-i} S_j(i-1) \end{aligned} \quad (15.31)$$

其中， t 的范围是整个情节，两个规则活动。换句话说， R_i 的强度跟随 R_j 的强度。如果 S_j 收敛到一个常数 S_j^* ，则 S_i 也收敛：

$$S_i^* = \lim_{t \rightarrow \infty} S_i(t) = \lim_{t \rightarrow \infty} [(1-b)^t S_i(0) + \sum_{i=1}^t (1-b)^{t-i} S_j(i-1)] = S_j^* \quad (15.32)$$

事实上，同样理由导致式(15.29)，表明 S_i 收敛到 S_j^* (the internal payoff for R_i)，其收敛速率与规则强度收敛到 PSP 外部奖励的预测值。这种分析可以扩展到任何规则链，例如 $\langle R_1, R_2, \dots, R_n \rangle$ 规则循序点火，仅 R_n 接收外部奖励。在平衡条件下，桶链算法使链中的全部规则具有相同的强度。在不平衡情况下，链中规则强度由式(15.31)描述，而不是式(15.32)。在任何情况下，如果链所接收的期望奖励相当一致，则链中规则的强度趋向收敛到一般水平。

为了比较 PSP 和 BBA，让我们考虑图 15.11，它给出 10 个状态，包括初始状态 A 和 B，结束状态 H，I 和 J。在状态 H，I 和 J 产生的外部奖励分别是 1000，0 和 300。

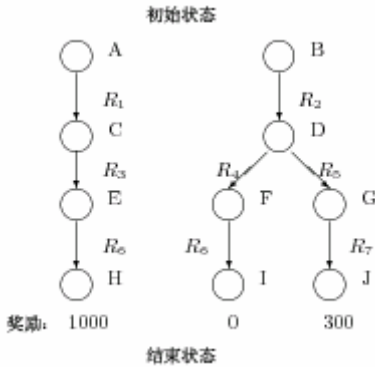


图 15.11 PSP 与 BBA 比较

分类器系统在这个例子下运行，使用 PSP 或 BBA 修改规则强度。全部初始强度设置为 100，投标比率 b 为 0.1，和 1000 情节在外部奖励执行划分，即每个分类器系统为 3000 步。得到的规则强度示于表 15.2 中。

表 15.2 不同的强度修改方案

规则	PSP 强度	BBA 强度
R_1	1000	648
R_2	299	567
R_3	1000	645
R_4	4	644
R_5	300	300
R_6	999	531
R_7	300	300

15.10 进化策略

进化策略模仿自然进化原理作为一种求解参数优化问题的方法。早期的进化处理是基于由一个个体组成的群体和一个操作符：突变。进化策略强调在个体级上的行为变化。最简单的实现方法如下：

- (1) 定义的问题是寻找 n 维的实数向量 \mathbf{x} ，它使函数 $F(\mathbf{x}) : \mathbf{R}^n \rightarrow \mathbf{R}$ 。
- (2) 双亲向量的初始群体从每维可行范围内随机选择。
- (3) 子孙向量的创建是从每个双亲向量加上零均方差高斯随机变量。
- (4) 根据最小误差选择向量为下一代新的双亲。
- (5) 向量的标准偏差保持不变，或者没有可用的计算方法，那么处理结束。

15.11 进化规划

进化规划(evolutionary programming)的过程，可理解为从所有可能的计算机程序形成的空间中，搜索有高的适应值的计算机程序个体，在进化规划中，几百或几千个计算机程序参与遗传进化。进化规划最早由美国的 L. J. Fogel、A. J. Owens 和 M. J. Walsh 在 1962 年提出。

进化规划强调物种行为的变化。进化规划地表示自然面向任务级。一旦选定一种适应性表示，就可以定义依赖于表示的变异操作，在具体的双亲行为上创建子孙。

进化规划最初由一随机产生的计算机程序群体开始，这些计算机程序由适合于问题空间领域的函数所组成，这样的函数可以是标准的算术运算函数，标准的编程操作，逻辑函数或由领域指定的函数群体中每个计算机程序个体是用适应值测度来评价的，该适应值与特定的问题领域有关。

进化规划可繁殖出新的计算机程序以解决问题，它分三个步骤：

- (1) 产生出初始群体，它由关于问题(计算机程序)的函数随机组合而成。
- (2) 迭代完成下述子步骤，直至满足选种标准为止：
 - ① 执行群体中的每个程序，根据它解决问题的能力，给它指定一个适应值
 - ② 应用变异等操作创造新的计算机程序群体。基于适应值根据概率从群体中选出一个计算机程序个体，然后用合适的操作作用于该计算机程序个体。把现有的计算机程序复制到新的群体中。通过遗传随机重组两个现有的程序，创造出新的计算机程序个体。
- (3) 在后代中适应值最高的计算机程序个体被指定为进化程序设计的结果。这一结果可能是问题的解或近似解。

习 题

1. 请简述进化系统的理论模型。
1. 请给出达尔文进化算法。
1. 叙述基本遗传算法的定义，并说明基本遗传算法的构成要素。
1. 遗传算法试图支持搜索时的遗传多样性，同时又要保持重要特征(用遗传模式表示)的存活，描述一种不同的遗传算子，它同时支持这两个目标。
1. 什么是简单变异操作，简述简单变异操作的过程。

1. 什么是简单反转操作，简述简单反转操作的步骤。
1. 考虑巡回推销员问题。讨论选择这个问题的合适编码问题。设计针对这个问题的其他合适的遗传算子和适应度度量。
1. 设计遗传算法搜索巡回推销员问题的解。
1. 请简述遗传算法种的染色体和基因是如何表示的。
1. 给出两个父个体 1110###0#和 1##0111##，设定交叉点为 6，请使用单点一直交叉杂交算子，写出它们的子个体。
1. 请用程序设计语言编写并行遗传算法。

第十六章 人工生命

人工生命是指用计算机和精密机械等生成或构造表现自然生命系统行为特点的仿真系统或模型系统。自然生命系统的行为特点表现为自组织、自修复、自复制的基本性质，以及形成这些性质的混沌动力学、环境适应和进化。

16.1 引言

在现实世界中，普遍地存在着各类复杂系统，例如人体系统、神经系统、免疫系统、心理活动、认知活动等都是复杂系统。一般认为，非线性、不稳定性、不确定性是造成复杂性的根源。

日常生活中的“流行”便是一种这样的现象：集团中的组成成员各人都穿着符合其个性的服装。可是有一天某种类型的服装呈现出优势，就会影响整个集团服装的倾向。整体倾向又改变了各个组成成员的状态，反过来又对整体发生影响。这样，微观的行为决定宏观的秩序，而宏观的秩序又左右微观的行为。同样，神经系统中的信息处理也是宏观与微观之间作用的过程。神经细胞是组成成员，神经网络是社会。当从外部得到新的信息时，脑一方面利用已有的信息，一方面吸收新的信息，自律地生成新的信息结构。每一个神经细胞都将影响神经网络，而神经网络又控制各个神经细胞的行为。我们理解新知识的过程，就是采用这种方式。从信息处理动态变化过程的观点看，“流行”与神经系统从本质上说都是一样的。这些复杂性的研究不能采用原有的要素还原法，即将各种事物细分成组成它的要素再加以分析研究。复杂事物只能照它复杂的面貌来理解。

美国圣菲研究所非线性研究组的兰顿(C. G. Langton) 于 1987 年提出人工生命(artificial life)。人工生命的独立研究领域的地位已被国际学术界所承认。在 1994 年创刊并在世界著名学府麻省理工学院出版的国际刊物《人工生命》(Artificial Life)，是该研究领域内的权威刊物。人工生命学术界举办的 7 次国际学术会议简要评述如下：

(1) “人工生命——关于生命系统合成与模拟的跨学科研讨会”。本次会议于 1987 年 9 月在美国新墨西哥的罗斯阿拉莫斯举行。本次会议的论文集共收录了 24 篇论文，内容主要分布在：人工生命研究的理论、生命现象的仿真、细胞自动机（简称 CA）、遗传算法、进化仿真等 5 个方面，兰顿发表了题为“人工生命”的开拓性论文，他在文中提出了人工生命的概念，并讨论了它作为一门新兴的研究领域或学科存在的意义。兰顿被公认为人工生命研究的创立者。这次会议标志着人工生命研究领域的诞生。

(2) “人工生命 II——人工生命研讨会”。本次会议于 1990 年 2 月在美国新墨西哥的圣菲举行。该会议论文集共收录了 31 篇论文，内容分为概貌、自组织、进化动力学、开发、

学习与进化、计算、哲学与突现、未来等 8 部分。其中兰顿的“混沌边缘的生命”、约翰·科赞（John Koza）的“遗传进化和计算机程序的共进化”属于经典之作。

（3）“人工生命 III——人工生命研讨会”，1992 年 6 月在美国新墨西哥的圣菲举行。本次会议的论文集共收录了 26 篇论文，内容除涉及遗传算法、进化仿真、突现行为、适应度地貌图、群体动力学和混沌机制等人工生命经典内容之外，还讨论了机器人规划应用问题。科赞的“人工生命：自我复制的自发突现与进化的自改进计算机程序”堪称杰作，从遗传编程算法方面探讨了在人工生命研究中关键的突现机理。

（4）“人工生命 IV——第四届国际生命系统合成与模拟研讨会”，1994 年 7 月在美国麻省理工学院举行。本次会议的论文集共收录了 56 篇论文，内容分为特邀报告、长文和短文 3 个部分，它覆盖了协同进化、遗传算子、进化与其它方法（如神经网络等）的综合、AL 算法、关于混沌边缘和分岔的研究、AL 建模、学习能力、进化动力学、细胞自动机、DNA 非均衡学说研究、人工生命在字符识别、机器人等方面的应用等较为广泛的内容。

（5）“人工生命 V——第五届国际生命系统合成与模拟研讨会”，1996 年 5 月 16-18，在日本古城奈良举行，来自世界各地的 500 多名学者参加了会议。这是人工生命首次在亚洲召开的国际会议。人工生命概念刚提出，就引起日本学者的关注，第一次人工生命国际会议就有日本学者参加。这次会议在日本的召开，标志着日本成为亚洲人工生命研究的中心。

（6）“人工生命 VI——第六届国际人工生命研讨会”，1998 年 6 月 26-29 在美国洛杉矶加利福尼亚大学举行。这次会议的主题是“生命和计算：变化着的边界”。本次会议收到大约 100 篇提交的论文，其中 39 篇作为完整论文在这次会议的论文集中得到介绍。有 9 篇论文被认为是人工生命的新的质量工作。这次会议主要的论文涉及的是计算的分子和细胞生物学。会议提供了许多新的关于发育过程、细胞分化机理和免疫反应模型制造的新见解。这些论文把人工生命扩展到令人兴奋的新方向。

（7）“人工生命 VII——回顾过去，展望未来”，于 2000 年 8 月 1-6 日在美国波特兰的里德学院举行。本次会议的主题是：“回顾过去，展望未来”。具体讨论的问题有以下几个方面：生命的起源、自组织和自复制问题，包括人工化学进化、自催化系统、虚拟新陈代谢等；发育和分化问题，包括人工的和自然的形态发生，多细胞分化与生物进化，基因调节网络等；进化和适应动力学问题，包括人工进化生态学，可进化性及其对生物组织的影响，进化计算等；机器人和智能主体，包括进化机器人，自主适应机器人和软件智能体等；通讯、协作和集体行为，包括突现集体行为，通讯和协作的进化，语言系统、社会系统、经济系统和社会-技术系统等；人工生命技术和方法的应用，包括工业和商业的应用，可进化硬件、自修复硬件和分子计算，金融和经济学，计算机游戏，医疗应用，教育应用等；认识论和方法论基础问题，包括人工生命的本体论、认识论以及伦理和社会影响等。

（8）“人工生命 VIII”于 2002 年 12 月 9-13 日在澳大利亚悉尼新南威尔士大学举行。本次会议具体讨论的问题有以下几个方面：生命起源，发育和差异，进化和自适应动力学，机

机器人和智能主体，通讯、协作和集体行为，人工生命技术的应用，仿真和综合工具和方法论等。

第9届会议人工生命国际会议于2004年9月12-15日在美国波士顿举行。2001年成立了国际人工生命学会(<http://www.alife.org/> International Society for Artificial Life)。2005年9月在英国肯特大学举行欧洲第8届人工生命学术会议，请参见网址：<http://www.ecal2005.org/>。

为什么要研究人工生命？在这一领域研究时要支持哪些东西。从控制我们的生态环境的工程新应用到在自然界中为我们提供较好的前景这个广阔的范围，都可以找到它的应用。

人工生命的研究可使我们更好地理解突发特征，个体在低级组织中的集合，通过我们的相互作用，常可产生特征。该特征不仅仅是个体的重叠，而且是总体上新出现特征。这样的现象可见于自然界的所有领域，但在生命系统中更为明显。生命本身确实有突发性质，当总体分解为它们的组成部分时，相互作用所产生的突发性质将全部消失。归约科学，它的研究方法今天看来是最严肃的学术研究，其中大部分是分析的方法。归约科学在各个领域都已取得很大成功。但自然的很多特性都被忽略，这并不是因为这些特性是无趣的或不重要的，相反，人们研究这些特性，但缺乏适当的工具和有效的方法来研究，人工生命领域的研究必须是综合的，把所有的因素综合考虑以创造生命形式，而不是肢解。

有很多方法使得新的特性可能突发，虽然，通过自然选择的进化，增加了复杂性，使这些方式不易得到和难以应用。在研究人工生命中共同的工具和方法很可能对其它领域也是有用的，许多杰出的物理学家研究突发性质已引起人工生命界的注意，对于研究自然界中的突发性质，人工生命提供了一种非传统的途径。

人工生命将会成为研究生物的一个特别有用的工具。对于研究像熵条件这样的抽象课题，可以支持有关生命问题的研究。长期以来，从有机体的简单模型到研究复杂现象，对生物方面的研究已取得了一定的效果。

人工生命不仅对于实验提供了短的生命周期(对于人是20年，果蝇为10天，Hillis分类算法为1秒)，人工生命系统对于控制和繁殖力提供了非并行的机会。人工生命也提供了机会使你看到有趣的但不可预测的突发性质，然后，返回以前的时间以研究它们的遗传。

人工生命作为生物界的基本研究工具，有能力使有机体和社会变得简单化，虽然是人工的，但也是有用的。

对于发展新技术及增强我们控制自然的能力，人工生命系统是很有潜力的。从任务的规范说明显式地产生代码。例如自动编程，用规范说明语言描述任务，然后精练所得到的程序，以改善程序系统的性能。也可以进行机器学习的研究，通过提供所希望的输入输出数据，训练计算机，获取所需的知识。

人工生命程序的开发首先选择一种计算体系结构，然后提供一系列的环境以使程序进化，使该程序可完成指定的任务。

目前遗传算法学术界已提供进化参数集，对于很多任务，该参数集可提供好的，但不必

是最好的解。

把这些能力工程化的最好方法无疑依赖于任务及现代的计算机体系结构技术。正像人工神经网络，通过学习获取新的能力，在本质上是并行的。一些大规模并行计算机系统在处理这些高度并行的问题时，取得了良好的效果。

人工生命的另一显著应用是遗传工程。对于传统的生命不能居住的地方，进化的新的生命有能力在这些地方居住。通过分子生物学的技术，具有这样的能力的可能性是很令人兴奋的。

过去很少重视，现在广为重视的是发展新的教育工具，要使下一代掌握更多的科学和数学知识是时代的需要。通过综合新的生命形式，使学生更容易理解和学习生物的突发性质。这种现象在社会科学中也是广泛存在的。

上面已提到过人工生命模型在生态学仿真和进化上是有效的。每年有数以百万的儿童死于昆虫传染病，世界上相当一部分谷物由于虫灾而歉收，若没有可用的化学农药，则将会造成很大的损失。过度使用农药也会导致世界生态系统出现非常严重问题。迫切需要找出办法更有效和更聪明地使用农药，这要求应能得到更好的模拟技术。国际上已开发人工生命软件模拟昆虫繁殖，以便用于蚊子控制，用于农业生产。在不同繁殖场所的昆虫，作为不同子昆虫群体，该昆虫群体可以繁殖，也可通过迁移和其它子昆虫群体发生作用，也可由于各种原因(旱灾、水灾)而消失。我们能够再生这些昆虫过去的群体周期，且能用不同方法控制它们。

16.2 人工生命的探索

长期以来，人类一直企图用当时的技术模拟自然界的工作，用人工办法理解自然界的本质。早期的机械技术提供工具，扩展人的体力劳动的本领，极大地减轻了人的体力劳动。20世纪初，逻辑在算术机械运算中的运用，导致过程的抽象形式化。丘奇(Church)、克里恩(Kleen)、图灵(Turing)和帕斯特(Post)形式化逻辑序列步的概念，形成了实现机械处理的基础。1945年1月6日在普林斯顿研究所有关脑和计算机的讨论会上，认为工程和神经网络是基础之一。后来，维纳(Wiener)提出了控制论。麦科洛奇和皮兹提出了M-P神经网络模型。冯·诺伊曼的兴趣是研究脑和计算机在组织上的相似性，研究用形式逻辑如何表示脑。1946年3月的控制论会议上，形成了以冯·诺伊曼为首的形式理论派和以维纳为首的控制论派。冯·诺伊曼方法将全部表示和演算还原到基本的逻辑世界，对符号用显式的逻辑过程实现。冯·诺伊曼计算机体系结构就是典型的代表。维纳开始使用信息、反馈、控制等概念，将生物和机械的问题在统一的科学概念下研究，形成控制论学派。

40年代末，50年代初，冯·诺伊曼提出了机器自增长的可能性理论。以计算机为工具，迎来了信息科学的发展。1956年在达特默斯的夏季讨论会上，麦卡锡提出人工智能这一术语。正式形成人工智能学科的研究。人工生命许多早期的工作也源于人工智能。60年代，罗森勃拉特(Rosenblatt)研究感知机，斯塔勒(Stahl)建立了一些细胞活动的模型，他把图灵机用作

“算法酶”，将生化表示成字符串。60 年代后期，林登麦伊尔(Lindenmayer)提出了生长发展中的细胞交互作用的数学模型，现在称为 L-系统。这些相当简单的模型，可以明显地显示复杂的发展历史，支持细胞间的通信和差异。

70 年代以来，科拉德(Conrad)和他的同事研究人工仿生系统中的自适应、进化和群体动力学，提出了不断完善的“人工世界”模型。后来侧重研究系统突发性的个体适应性。乔姆斯基(Chomsky)的形式语言理论应用在程序设计语言的规范说明和开发编译程序。细胞自动机应用于图象处理。科伟(Conway)提出生命的细胞自动机对策论。在这种理论中，如果一个细胞周围的 8 个细胞中有 3 个细胞正好是通，那么该细胞开通，若周围细胞有 2 个或 3 个通，那么它就保持通，否则它就关断。这样的细胞自动机系统已被广泛试验。

与人工智能有关的领域，如经典逻辑、搜索算法、启发式搜索、推理、神经网络、遗传算法、模糊数学等都取得了很大进展。其中最重要的是提出以知识为基础的推理，即知识工程、专家系统，使人工智能进入实用，各种专家系统纷纷出现。但是，专家系统在实际应用中暴露出了不少问题，诸如知识表示、常识性知识、非单调推理、框架问题、知识获取、解题的有效性等。

80 年代，人工神经网络又兴起，出现了许多神经网络模型和学习算法。与此同时，人工生命的研究也逐渐兴起。1987 年召开了第一届国际人工生命会议。

16.3 人工生命模型

当前，人工生命的研究现状与人工智能早期历史可以说是并行的。现在已有许多人工生命程序的例子，著名的有：

1. 计算机病毒

计算机病毒提供了人工生命生动的例子，描述了生命系统所固有的特征——繁殖、机体集成，不可预见性等。像病毒一样的程序，或许对计算机网络有用。

2. 计算机的进程

类似于计算机病毒，它把进程当作生命体，它在时间空间中可以繁殖，从环境中汲取信息，修改所在的环境。这里应当加以区分，不是说计算机是生命体，而是说进程是生命体。该进程与物质媒体交互作用以支持这些物质媒体(如处理器，内存，当有虚拟存储时包括磁盘)，可把进程认为具有生命的特征。

显然，计算机进程与自然生命有本质的区别，因为进程间的连接和支持进程存在的物质有很少的内在联系，例如，占有 CPU 的计算机进程可被解释并被送至内存，或送到磁盘，与此同时，其他的进程在 CPU 上执行，不同之点是表面的。

一些种子保持冬眠达数千年，在冬眠期内既没有新陈代谢，也没有受到刺激，但毫无疑问，它们是有生命的，在适当的条件下即可发芽。类似地，计算机进程也可在内存某个地方之外活着，正等待适当的条件重新出现以便恢复它们的活动状态。

3. 生物统计学和个体胎生学

生命系统能进化和具有产生复杂的集成模式的形态的能力。自然界中这种能力看来几乎是唯一的、复杂而美丽的模式，常常有些分形几何图形，可由生命细胞的简单动作的重复及综合结果而出现。

当前，类似于生命形态的系统的进化技术与自繁殖能力是有差别的。至于我们的理解力及通过进化位串发展而产生复杂的表现型能力。计算机图形的应用将变得更为广泛，故我们看到有潜力产生美丽的、类生命的、不能预料的模式和行为，有人已经编程出进化神经网络，网络中个体有举止，可以移动，对另一些个体发送信号。更为强调的是要研究基因和整个机体之间的事件链结，遗传型正好是位串，这些位串的参数正像人工神经网络中的权值或阈值。

4. 机器人

机器人提供了类似于生命系统的完全不同途径。从形态上看与生物相似，然而它没有繁殖能力，但它们可以通过推理预见将来。机器人有很多性质与生命有关——复杂性、机体集成、受刺激、及可移动。

5. 自催化(autocatalytic)网络

它与高技术的机器人相反，由人设计和构造，在模拟化学试管中产生类似生命的可自繁殖(或自进化)的简单系统，这类系统有两类正处于活跃的研究阶段——自催化网络和细胞自动机。

自催化网络有很多种变形，发展最完善的是超周期理论(theory of hypercycle)。它被连到反应元件(RNA)网络，允许连贯的功能偶联进化、自复制实体。这里利用超周期网络为例子来讲述所有的自然生命系统，Eigen 已开发了关于超周期的推理理论，它可自发增加它的复杂性，以及什么时候会崩溃。它的数学处理是基于连续反应运动学的。假设元素的完全混合没有空间的结构，它优于细胞自动机模型。

6. 细胞自动机

它是细胞阵列，每个细胞是离散结构，根据预先规定的规则，这些状态可随时间而变化，传递规则是通过阵列，计算每个细胞的当前状态以及它的近邻的状态，所有的细胞均自发地更新。自催化网络和细胞自动机都产生不随时间而变化的自组织、自复制模式，但处理它们的数学模式是不同的。

细胞自动机的历史是辉煌的。1940 年它由冯·诺伊曼发明的，可说是用数学和逻辑形式提供了理解自然系统(自然自动机)的一种重要的方法，它也是理解模拟和数字计算机(人工自动机)的一种系统理论。随着大规模并行单指令多数据流(SIMD)计算机的发展，很容易获得低价格的彩色图象，使得细胞自动机的研究更为方便。兰顿的工作可辨别细胞自动机规则集，可支持自组织过程及自然系统中的位变化，这为实现冯·诺伊曼计算机视觉迈出了一步。在前苏联，Vladimir kuz'min 和他的同事正朝着另一个方向使用细胞自动机模拟生命。由于细胞自动机和计算机的紧密联系，为研究图灵机等价于某类化学系统和生命细胞提供了桥梁。

Boerlijst 和 Hogeweg 将超周期和细胞自动机两种方法结合起来产生自繁殖过程，对于

超周期理论问题提供了令人满意的解，并说明自然系统怎样组织起来及怎样自发增加复杂性。

7. 人工核苷酸

人工生命并不局限于计算机，认识这一点是很重要的。许多被酶作用的物质可以支持生命，化学系统所形成的各种生命正在开发。

1960 年 Sol Spiegelhe 和他的同事结合当时已知的分子的最小集合，允许在一个试管中进行 RNA 的自复制——核苷酸前体、无机物分子、能源、复制酶、以及来自 α β 细菌噬菌体 (bacteriophage) 的 RNA 的雏形。满足了这些需要后，细菌噬菌体 RNA 分子不再需要感染的细菌宿主，但可以很快地复制以保持合适的频率，一系列转移到 RNA 分子的数目快速增长，但与此同时 RNA 分子本身反而变小了，直到达到最小的尺寸为止。分子群体从大量的、易传染的形式变为小的、不易传染的形式，大抵是因为它可从细菌核苷酸处脱落。很清楚，这些复制和进化 RNA 分子是同原始人工生命形式类似的。

分子生物学的新发展已经变得更加有趣，促使人工分子进化。Cech 和 Allman 发现 RNA 有酶化学和复制能力，这是非常关键的。这允许单个 RNA 分子进化。另一有意义的发展是扩张 DNA 的聚合链反应。

16.4 人工生命的研究方法

按照人工生命的组织机构，人工生命的内容大致可以分成两类：

(1) 构成生物体的内部系统，包括脑、神经系统、内分泌系统、免疫系统、遗传系统、酶系统、代谢系统等。

(2) 在生物体及其群体中表现的外部系统。生物群体中环境适应系统和遗传进化系统等。

从生物体的内部和外部系统所得到的各种信息，构成人工生命研究的方法，主要有两类：

(1) 模型法。根据内部和外部系统所表现的生命行为，建造信息模型。

(2) 工作原理法。生命行为所显示的自律分散和非线性的行为，它的工作原理是混沌和分形，据此研究它的机理。

例如，现在认为，人工神经网络是生物内部系统的模型系统，而遗传算法是外部系统的模型系统。在神经网络的信息处理中，反应混沌的特点，而遗传算法认为是自律分散的并行处理。从这样的模型系统可以看到各种各样的生命固有的行为。

人工生命的研究必须将信息科学和生命科学结合起来，形成生命信息科学，这可以采取下列策略：

(1) 采用以计算机等信息处理机器为中心的硬件生成生命行为。一般有两种方法：一种是采用已有的信息处理机器和执行装置，实现具有人工生命行为的系统。另一种是用生物器件构造生命系统。这些都通称为生物计算机，是一种向人工生命接近的方法。

(2) 用计算机仿真，研究开发显示生命体特征行为的模型软件。简单地说，神经网络系

统和遗传算法等，都是采用信息数学模型，模拟人工生命的生成。

(3) 基于工作原理，利用计算机仿真生成生命体。生命现象的基础是随物理熵的增大而杂乱无章。生成这种现象的原理是混沌的分形、耗散结构、协同反应等，采用这些产生生命现象。

(4) 通过计算机仿真，分析生命特有的行为生成，建立新的理论。利用上面 3 个策略，得到生命行为共同的一般性质，通过概括，建立生命的基本理论。这种策略形成自组织、超并行处理等理论。

上述 4 种人工生命的研究策略，基本上是从真实的生物信息处理系统接受许多概念和启发，用计算机等人工媒体生成构造生命。将真实生物的生命概念抽象、泛化、扩充。人工生命研究的成果将以还原主义的分子生物学为中心，可以作为现代生物学的补充。抽象、泛化、扩充的生命行为信息处理机制可以为精密机械、控制工程、计算机科学等其它领域所用。人工生命是形成新的信息处理体系强大的推动力。

人工生命的目标之一是扩充用于创建理论的经验数据库。一种办法是在计算机内创建进程，而其他可展现类生命行为的人工媒体，没有任何显式仿真，也没有任何具体已知的生物体模型。

冯·诺伊曼研究与构造结构有关的自繁殖，他发现可以采用两种不同的方式进行机体的遗传描述：

- (1) 解释指令而构造子孙，
- (2) 不解释数据而只是把它复制到创建的描述副本，然后传给它的子孙。

当 Watson 和 Crick 揭示 DNA 分子的工作机理时，表明这是生物体的遗传情况。

冯·诺伊曼给出了机器自复制可能性的构造性证明，他要求自复制机器具有计算普遍性和构造普遍性。Smith 也给出了复制普遍性的证明。他的证明优美地应用了递归函数论的著名递归定理。

关于计算机的另一种生命形式就是计算机病毒，它已存在 10 年之久，越来越多的计算机病毒迅速发展，不久的将来会成为传染性的。生物学家争辩着生物病毒是否有生命。尽管它们可以自繁殖，但它们没有自己的新陈代谢以复制需要的部分，而必须接过母体的新陈代谢以繁殖后代，由于媒体不同，计算机病毒看来比生物病毒多少活跃一点。

研究人工生命的目标就是试图追寻最高科学的和智能的目标，但如不小心保管，让他们随便钻进计算机网络，或钻进生物圈，则将会对现实世界造成毁灭性的后果。现阶段已知道人工生命会带来少量危险，随着我们对生物合成现象有越来越多的了解，这种危险将会增加。当机器人能够自繁殖，正像产生自繁殖的计算机那么容易时，不知世界将会变成什么样。

人工生命为讨论大量哲学问题提供了一个新的论坛，这些是有关生命的本质、智能和所存在的经典的和现代的问题。关于人工智能的可能性有很多争论，对于“思维”和“知觉”这样的概念好像陷入毫无希望的泥潭之中。

对于人工生命的争辩多少有点像对于机器智能的争辩，人工智能和人工生命都试图在计

计算机上实现。在科学家思考什么是“智能”时，人工智能已经完全改变了它的方向。虽然人工智能没有达到它的总目标，但它已对科学界做出了很多贡献。

与人工智能类似，人工生命可以促使我们思考什么是“活着的”。事实上，对于“生命状态”没有公认的定义。当问及什么是生命时，生物学家经常会列举出大多数生命系统的主要行为和特征，如：自繁殖、新陈代谢、死亡、复杂组织和行为等等。诚然，以上所列的都组成了严格的行为标准，但计算机进程也有上述行为。对此，我们有两种选择。

- (1) 承认计算机的行为是活着的，
- (2) 改变以上所列的行为标准，使得计算机进程不是活着的。

人工智能在达到“智能”方面失败的原因是它试图追逐一个活动的目标。普遍认为下棋需要真正的智能，现在计算机下棋可击败大多数人，故有人认为不能把下棋的能力作为智能的标志。

正如人工智能的情形，对于人工生命模型的本体性状态有两种意见：一种是弱解释：计算机进程只不过是生命的仿真。当然这很清楚，这在讲述“真实”生命某些具体科学问题时是有用的，但进程从来不认为是生命本身的例子。另一方面是强的解释：以上所列的行为已包括了所有已知的生物生命，同时也包括了某些计算机进程。因而应当考虑进程实际上是“活着的”。

首先必须区分定义生命的质量和那些生命的简单特性。很多概念对于思考和日常谈话是有用的，但要精确地定义是不可能的，例如，众所周知的“机器”，“游戏”，以及“爱”，但是“活的”则是另一回事。

根据活力理论，活的系统与不活的系统有本质的区别。这使生物学家努力辨识活的生命体和不活的，今天少数生物学家仍认为这是非常值得重视的问题。虽然自然生命系统的几乎所有特性能够被辨认，但对于生命还没有令人满意的形式定义。生命系统中的一些最重要的特性是：

- (1) 生命体的复杂性和组织；
- (2) 构成它们的分子的化学结构唯一性；
- (3) 单个生命体的唯一性及变异；
- (4) 拥有遗传程序，最终形成遗传表现型；
- (5) 通过自然选择形成；
- (6) 行为的明显不确定性。

以上的确有助于理解是什么使得生命体是唯一的。

关键问题是自繁殖，它是大多数天然生成的有机体的最重要的特征，同时也是区分生命体与非生命体的最重要特征。但繁殖的能力并不是所有的自然生命体都具有的，生命的界线仍被认为是活着的，如锯螻(prions)含有蛋白质，但不含核糖核酸，又如爱滋病反转录病毒(AIDS retroviruses)含有RNA，但没有DNA，它们自身不能繁殖，它们取代宿主的代谢系统。繁殖与生命定义的关系也不明确，例如骡子，阉人，它们能走，谈话，思考，从所有的标准

信号 06 在连接点处的结合变化如下，可以看出数据路径的扩张。

2	2	2	2	2	2	2	2	2		2	2	2	2	2	2	2	2	2
1	1	0	6	1	1	1	1	1	→	1	1	1	0	7	1	1	1	1
2	2	2	2	6	2	2	2	2		2	2	2	2	0	2	2	2	2
*	*	*	2	0	2	*	*	*		*	*	*	2	1	2	*	*	*
*	*	*	2	1	2	*	*	*		*	*	*	2	1	2	*	*	*
*	*	*	2	1	2	*	*	*		*	*	*	2	1	2	*	*	*

我们构造下述变换规则表，由此即可构造出二维空间中的细胞自动机的生长。

$f(0, 1, 2, 7, 6)=1$	$f(7, 0, 0, 0, 2)=3$	$f(2, 0, 0, 2, 3)=7$	$f(1, 0, 2, 3, 2)=6$
$f(0, 1, 2, 3, 2)=1$	$f(3, 0, 2, 2, 1)=0$	$f(7, 0, 2, 1, 2)=0$	$f(1, 0, 7, 2, 2)=3$
$f(4, 0, 2, 0, 2)=2$	$f(2, 0, 0, 2, 4)=0$	$f(2, 0, 2, 6, 2)=4$	$f(2, 0, 0, 1, 4)=2$
$f(4, 0, 2, 6, 2)=2$	$f(2, 0, 4, 6, 2)=4$	$f(1, 2, 4, 2, 6)=4$	$f(4, 1, 2, 2, 2)=0$
$f(2, 0, 0, 4, 2)=0$	$f(2, 0, 2, 4, 2)=0$	$f(2, 0, 1, 2, 4)=2$	$f(6, 0, 2, 4, 2)=4$
$f(7, 0, 0, 2, 1)=0$	$f(0, 1, 2, 7, 2)=4$		

上表中函数 f 的第一个自变量为中心细胞的状态，接着的 4 个自变量是 4 个邻居的状态顺时针方向旋转，以便产生最小的 4 位数。

细胞自动机的概念可由以下方式建立：首先我们有一细胞空间，它组成了 N 维欧几里德空间，以及定义于该细胞空间的邻居关系。对于每个细胞空间，由于邻居关系，必有有限个细胞作为它的邻居。一个细胞自动机系统(简称“细胞系统”)是这样定义的：该系统对每个细胞给定有限个状态和一个区分状态(叫“空状态”)，及一条规则。该规则给出每个细胞在时刻 $T+1$ 时的状态，且该规则是在时间 t 时该细胞自身的状态及它的邻居的状态的函数。我们把一个细胞的所有可能状态连同管理该细胞的状态变换的规则一起称为一个变换函数。所以一个细胞自动机系统是由一个细胞空间和定义于该空间上的变换函数所组成。细胞自动机状态由有限个细胞连同赋予每个细胞的状态所指定，可理解为其它的细胞都处于空的状态。

以下我们讨论细胞自动机的问题。

为避免讨论边界问题，我们考虑一无界的空间，它是 N 维欧几里德空间被分为相等大小的正方形的细胞，好像图纸上的方格，我们称该空间为格状空间(tessellation)一个给定的细胞的邻居是指每一个坐标与给定的坐标至多相差 1 的所有细胞。格状结构可形式定义为五元组 (N, T, S, q, f) 。这里 N 是正整数， T 是 N 维欧几里德空间的子划分，即划分为细胞集，每个细胞都是边长为 1 的 N 维方体， T 的中心是正整数坐标。 S 是由状态组成的有限集， q 是 S 的区分状态，称之为静止状态。 f 是映射函数，它把任一个细胞 X 在时刻 $T-1$ 时的邻居 3^N 个细胞的所有状态的集合映射到 X 于时刻 T 时的状态。

格状细胞的有限方块叫做阵列，这种阵列的状态或构型是与每个细胞的状态相关的函数。也就是说一个阵列构型规定了在时刻 T 时阵列的每个细胞的状态。如果 C' 阵列满影映射到 C

阵列，同时每个细胞与它的映像有同样的状态，则构型 C' 是构型 C 的一个复制。如果存在 C^* 阵列的 N 个两两不相交的子集，其中每个子集都为 C 的一个复制，则构型 C^* 含有构型 C 的 n 个复制。

16.6 形态形成理论

典型的形态形成理论是 1968 年 Lindenmayer 提出的 L-系统。L-系统由一组符号串的重写规则组成，它与乔姆斯基(Chomsky)形式语法有密切关系。在下面“ $X \rightarrow Y$ ”表示结构中当出现 X 时用字符串 Y 代替。因为字符 X 可以出现在规则的右边和左边，这组规则可以被递归地应用来重写新的结构。

这里是一个简单的 L-系统的例子。规则采用上下文无关，即在特定部分改变时不考虑上下文中的关系。例如一组规则：

- (1) $A \rightarrow CB$
- (2) $B \rightarrow A$
- (3) $C \rightarrow DA$
- (4) $D \rightarrow C$

当把这组规则用于初始种子结构“ A ”，就可以得到下面的序列：

次数	结构	应用规则
0	A	初始种子
1	C B	规则 1, CB 代替 A
2	D A A	规则 3, DA 代替 C; 规则 2, A 代替 B
3	C C B C B	规则 4, C 代替 D; 规则 1 用两次, CB 代替 A
4	...	继续进行

L-系统与元符号结合，可以表示分支点，允许从主干分支产生新的符号行。规定符号“()”“[]”分别表示左和右分支。假定有下列规则：

- (1) $A \rightarrow C[B]D$
- (2) $B \rightarrow A$
- (3) $C \rightarrow C$
- (4) $D \rightarrow C(E)A$
- (5) $E \rightarrow D$

当把这组规则用于初始字符串“ A ”，就可以得到下面的序列：

次数	结构	应用规则
0	A	开始种子
1	C[B]D	规则 1.
2	C[A]C(E)A	规则 3, 2, 4.

- 3 C[C[B]D]C(D)C[B]D 规则 3, 1, 3, 5, 1.
- 4 C[C[A]C(E)A]C(C(E)A)C(E)A 规则 3, 3, 2, 4, 3, 4, 3, 2, 3.

为了沿着结构传播信号，在规则左边具有多于 1 的符号。这时规则就成为上下文有关的。在下面的例子中，我们规定在“{ }”中的符号或符号串将被代替，而上下文左边其余的符号“[”和“]”分别表示字符串的左端和右端。例如，规则组包括下列规则：

- (1) [{C} → C 在字符串左端的“C”保持“C”
- (2) C{C} → C “C”和它左端的“C”保持“C”
- (3) *{C} → * “C”和它左端的“*”变成“*”
- (4) {*}C → C “*”和右端的“C”变成“C”
- (5) {*}] → * 在字符串右端的“*”保持“*”

在这些规则下，初始结构“*CCCCCC”将产生向右传播：

次数	结构
0	*CCCCCC
1	C*CCCCC
2	CC*CCCC
3	CCC*CCCC
4	CCCC*CCC
5	CCCCC*CC
6	CCCCCC*C
7	CCCCCCC*

信号传播能力非常重要，它允许任何计算过程嵌入在结构中，可以直接影响结构的发展。

16.7 混沌理论

从系统角度来看生命的行为，首先在物理上可以定义为非线性、非平衡的开放系统。所以，这种系统特殊行为是自然振荡，即有限周期和混沌。这些生命特有的行为是本体主义和高级信息处理所关注的，人工生命中，它的生成原理是主要的。有限周期可以分为稳定的和不稳定的。把自然振荡称为稳定的有限周期。这种稳定的有限周期邻近的位相面轨道渐近闭轨道。有限振荡和混沌是结构稳定的两个方面。它们的差别仅仅在于有限周期的初始条件的差对于以后运动影响很大。在混沌中，初始条件的差随着时间很快扩大，近似再现是困难的。这种性质称为轨道不稳定性，这和基于有限周期的轨道稳定性的有序化形成明显对照。自然界中动力学分为秩序和混沌。秩序的规则行为是周期解。其它混沌系统是复杂的，长期不能预测的行为。两者从时间发展上基本上是动力学系统。所以，生命体是混沌和秩序的复合。

生命行为所提取的特征现象的发生原理的秩序和混沌的计算能力怎样呢。首先，秩序系统具有在它的周期上的复杂性，高度的复杂计算是不可能的。混沌系统必须有严格的规定，

从理论上计算它的轨道的不稳定性行为是困难的，因此，仅考虑在混沌的边界所看到的生命行为。这样，信息的存储、传送、变换之类的基本操作可以通过秩序和混沌的混合系统实现：

(1) 信息存储是从有序系统所看到的稳定周期动力学获得。

(2) 信息的传送和变换可以从混沌系统不稳定动力学实现。

兰顿发现，通过调整描述细胞自动机规则中的一个参数 λ 就能让这些一维的细胞自动机连续的变化于固定值→周期型→复杂型→混沌型。因此，兰顿提出复杂型的细胞自动机是介于秩序（固定值、周期）与混沌之间的一种状态，他把它命名为“混沌的边缘”。进一步，人们不难发现，自然界中一切复杂的现象，包括生命、智能、社会实际上都是介于秩序与混沌的边缘。因为系统要想足够复杂，就既不能过于秩序而变得僵死，又不能过于混沌而变得混乱，系统会巧妙地自动平衡于秩序与混沌这两种状态之间。

16.8 人工生命的实验系统

人工生命的研究平台的目标是通过计算机对生命行为特征的模拟，可以最终形成生命计算理论。即计算机不会成为生命体，但可以作为研究人工生命的强有力工具，除了能表现出生命的一些基本行为，还能表现生命的一些特有行为，如自组织、自学习等。研究这些平台不仅有助于解释生命的全貌及探索生命的起源和进化，而且也为生物学研究提供了新的途径，同时也为人工生命的研究提供了有利的工具。

16.8.1 Tierra 数字生命进化模型

1991 年，美国俄克拉何马大学的生物学家汤姆斯·雷 (Thomas S. Ray) 完成了一个叫 Tierra (西班牙语，意为地球) 的计算机模拟程序。汤姆斯·雷是一位研究热带雨林的进化和生态问题的生物学家，在他开发的 Tierra 系统中，很多具有自复制能力的数字生物构成了一个虚拟生命世界，计算机的中央处理器和内存组成的物理环境使得演化过程得以进行。中央处理器的时间代表能量，内存空间代表资源，这些数字生物或自复制程序不断地改变自身的进化策略，在 Tierra 世界中为生存展开竞争。那些能够获得更多时间和内存空间的程序可以在下一代中留下更多的拷贝，反之则会被淘汰。该系统的运行展现了很多进化和变异特征，以及与地球生命相近的种种行为。刚开始时，Tierra 模型中只有一个简单的祖先“生物”，经过 526 万条指令的计算后，Tierra 模型中出现了 366 种大小不同的数字生物。经过 25.6 亿条指令后，演化出了 1180 种不同的数字生物，其中有些寄生在别的数字生物体内，有些对寄生生物具有免疫能力。Tierra 还演化出了间断平衡现象，甚至还出现了社会组织。总之，几乎自然演化过程中的所有特征，以及与地球生命相近的各类功能行为组织，都会出现在 Tierra 当中。Tierra 的主页是：<http://www.his.atr.jp/~ray/tierra/index.html>。

Tierra 的最可喜的成果是证明机器代码能够进化。这意味着机器代码能变异或再组合，并且产生的代码保留足够的功能以便通过自然选择能够不时地改变代码。另外，自然环境随

着有机体的进化而进化。这是推动地球上的生物物种和复杂度上的进化的基本因素之一。

Tierra 另外一个成果是将一个数字的多细胞的类似物加入 Tierra 中。汤姆斯·雷觉得多细胞是对并行计算的天然类推，是增加进化的丰富性的特征。多细胞 Tierra 模型中的特征是：

- (1) 细胞有机体以单细胞产生，通过一个二进制的细胞分裂过程发展成多细胞形式；
- (2) 细胞个体的每个细胞有相同的基因物质，因为它们来自于一个初始细胞；
- (3) 充分发展的不同细胞有潜在的不同性，以便它们能表示基因组的不同部分。

然而，多细胞的进化增加了细胞的数量，但没有增加细胞的类型。

为了在进化的多细胞数字有机体中引起细胞类型的增长，Tierra 运行在分布式的计算机网络上。汤姆斯·雷利用一个叫 Beagle 的新观测工具，观察在网络中运行的 Tierra 和控制网络中的 Tierra。汤姆斯·雷设计的数字生命以数字为载体，探索进化过程中所出现的各种现象、规律以及复杂系统的突现行为。

16.8.2 Avida

Avida 是一个自适应的遗传基因系统。Avida 的概念与 Tierra 的概念相似。Avida 明显的不同于传统的遗传算法。在 Avida 中的选择比大多数的遗传算法机制更接近于自然选择。

Avida 系统在计算机的内部产生一个人工的(虚拟的)环境。系统实现了一个虚拟的二维网格处理器，这个处理器运行一种有限的组合语言；程序作为指令的连续命令字串被储存在系统内存中。每个程序(生物体)与一个网格相关。

这个程序和它的派生物随后受到各种可能的随机突变的影响，任何在指定环境中导致繁衍能力提高的突变都被认为是有利的。绝大部分突变是不利的(典型的如引起生物无法完整地繁衍)或中性的，只有少数有利的突变使生物体更有效地繁殖，使之在此环境中茁壮成长。经过时间的洗礼，更加适应环境的生物体产生了。Aviria 系统中重要的概念如下：

(1) 分时

分时是一种分派给网格中的生物去运行它们的代码占用处理器时间的方法。一些生物比其他生物有更快速的处理器，所以分时代码一定要确定他们运行在适当的(相对的)速度。每个生物都有一个决定它的处理器速度的指标；时间分配器将会调整这个指标直到在这个环境中生物执行任务达到满意为止。分时机制深远地影响生物群体的全局行为。

(2) 适应度

适应度是在特定环境中的特定生物，对于它的复制能力的一个无单位测量法。对于本身来说，适应度没有多少内在意义，但是与其他生物比较时，适应度给出它们各自复制率的比。为了计算适应度，特别地用一个生物指标与准备时间相除。如果一个生物的适应度是另外一个生物的两倍，它将会在每个单位时间内比另一个生物有多一倍的后代数。适应度的刻画，使程序会发展到显示出有用的行为而不是简单的自我繁衍。

(3) 繁衍

自我复制产生后代。在 Avida 中的繁衍是通过四个不同的过程来完成的：

- a. 给子程序分配一个新内存;
- b. 逐条指令地把父程序拷贝进入新内存之中;
- c. 把程序分裂为父程序和子程序;
- d. 把子程序安置在格子中。

(4) 选择一个安置方法

后代产生后, 将后代安置在父体周围的哪个格子中呢? 有以下几个选项:

a. 选择空的位置: 这是一个非常有限的生育方法, 但当只有死亡被启动时, 它又是普遍见效的方法。在这个模式中, 细胞被禁止彼此杀害, 而且只有新生的生物被允许搬进空的细胞。

b. 选择最老的: 这在 Avida 中是默认的方法。在父体(包括父体本身)周围的生物将会被评估, 它们中间最老的将会被删除。在程度相同的情况下, 会在最老的单元中随机选择一个来删除, 以便新生物的到来。

c. 选择最大年龄/指标: 这个安置方法有利于有较高指标的生物, 而且有利于刺激生物学习额外的特殊技能, 而且这是鼓励生物学习特殊技能的附加方法。

d. 随机选择: 从父体和它的八个邻居中随机选择子生物。因为大约在他们有后代的任何机会之前, 大约半数的生物会被替代, 所以这种方法对进化来说是不好的。

(5) 突变

在 Avida 中实现的突变类型是:

- a. 拷贝突变: 在拷贝时发生的突变。
- b. 点突变: 随机发生的突变。
- c. 分裂突变: 父体与子体分裂时发生的突变。
- d. 分裂插入、分裂删除: 分裂时故意插入或删除某个字符的突变。

一般来说, 所有的突变率必须是在一个确定的界限下, 以避免杀害群体, 当比率太低时进化就会变得很慢。Avida 系统中一些参数变量可以修改配置, 使得 Avida 系统更接近于自然选择、适者生存的达尔文生物进化过程。

16.8.3 Terrarium 生物饲养生态系统

微软公司为了展示 .NET 框架中的一些重要特性和激发编程人员的学习兴趣, 推出了一个名为 Terrarium 的生态系统示例程序。Terrarium 实际上是一款向软件开发人员全面展示 .NET 框架应用开发技术的教学游戏, 同时也是一个学习进化生物学和人工智能的建模工具。在 Terrarium 游戏中, 开发人员可以创建草食动物、肉食动物或植物, 并将它们放到一个基于“适者生存”模型和对等网络结构的生态系统中。该游戏既提供了一个可以测试开发人员的软件开发与策略设计水平的竞争环境, 也提供了一个近乎真实的进化生物学和人工智能模型, 用以检验具有不同行为和属性的生物在生存斗争中的适应能力。

微软的生物饲养生态系统 Terrarium 的网址是

<http://www.terrariumgame.net/terrarium/>。微软的服务器负责运行整个生态圈，用可管理的容器方式让所有的生物在其中生活。在服务器端有一个环境程序，带有生物世界的一些基本参数：时间、能量、资源、空间等等。在客户端有一个种子程序，它是包含着底层规则的代码集，具备在环境程序中发育成一个完整数字生物的 DNA 代码。用户可以修改种子程序的源代码，使得发育出来的数字生物具有与众不同的能力与习性。种子程序分为植物、动物、微生物三类，分别可以通过修改发育成三类不同的数字生物。发育成的数字生物组成食物链，展开生存竞争。为了生存的需要，数字生物之间可以联合，表现为共生、寄生等现象，各类生物的种群规模最后达到一种相对平衡的状态。如果有新的物种加入进来，则可能打破平衡，进而展开新一轮的竞争。由于竞争永无止境，用户必须不断编出新的物种来接受生存挑战。环境程序的编程人员还可以引入陨石、洪水、干旱等灾难来对各个物种的生存能力进行破坏性实验。种子程序也不断升级，例如，在环境中加入了对水的定义，种子则必须相应地把水作为生存要素之一。因此，用户必须不断地跟着升级，否则将在每年一次的灾难日被灭绝。

16.8.4 人工鱼

涂晓媛在加拿大多伦多大学完成的博士学位论文题为《人工动物的计算机动画：生物力学、运动、感知和行为》，该论文 1996 年获得了美国计算学会 ACM 最佳博士论文奖。论文研究工作主要研究开发了人工鱼[Tu 1996]。

涂晓媛的“人工鱼”是一个栖息在虚拟海底世界中人工鱼群的社会。每条人工鱼都是一个自主的主体，既可以独立地活动，也可以相互交往。“人工鱼”有与鱼脑相对应的“意图发生器”，有与鱼眼对应的基于计算机视觉的虚拟感受器官，可以识别和感知其他人工鱼以及周围的虚拟海洋环境。每条鱼都以“感知-动作”模式生存，表现出包括自激发、自学习、自适应等智能特性，从而产生相应的智能行为。例如，因饥饿而激发寻食、进食行为；学习其他鱼的惨痛教训，不去吞食有钩的鱼饵；适应有鲨鱼的社会环境，逃避被扑食的危险等等。人工鱼群体是一种典型的多主体的分布智能系统，鱼群的社会表现出某些自组织能力和智能集群行为。例如，人工鱼群体在漫游中遇到障碍物时，能够识别障碍改变队形，绕过障碍后又重组队列继续前进。

16.8.5 AutoLife

张江研制的 Autolife 模型是一个能够进行“开放式进化”的人工生命系统[李建会 2006]。每个主体(agent)模型采用可以变化规则表长度的有限自动机模型建模。一方面主体可以进行自我繁殖，同时模型中的选择机制没有采用显式的适应度函数而是采用能量消耗的简单模型而自发涌现出来，所以可以认为主体模型是一个类 Tierra 系统。然而与 Tierra、Avida 等数字生命模型不同的是，Autolife 模型进行了大大的简化，它界面友好，操作直接。虽然没有给每个主体装配一个虚拟计算机，但是主体与环境的耦合则可以看成一个图灵机模型，因此主体可以通过变异而“任意”的编程序。通过 Autolife 模型，首先人们可以看到一般的

生态系统中共存的现象：生物的大爆炸、大灭绝，主体进化得越来越聪明；其次，用户可以通过变化不同的食物添加规则探索主体与环境的关系；最后，如果允许主体通过播种改变环境自动产生食物，那么组织的涌现就是一种不可避免的结果。主体构成的组织具有自主运动的特性，还可以进行自我修复，可以说 Autolife 中的组织是一些真正的“活体”。

1. 内部结构：一群参数包括拥有的能量值、寿命、当前坐标、当前面对方向等等。一个可变化长度的规则表 rules，这个规则表既可以当作指导生命体运动的程序规则，又可以当作遗传的数据（染色体）被遗传变异。

2. 如何行动：每个仿真周期，生命体仅仅读入它所面对的前方三个方格情况，如果 0 是空格，1 是有食物的方格，那么，每次生命体读入的就是一个 010, 001 这样的三位长的二进制字符串。生命体还有一些被编码成 0, 1, 2, 3, 4……的内部状态，这样根据读入的二进制串和内部状态，生命体会查找它的规则表 rules，找到匹配的一条规则，得到输出动作 (0, 1, 2, 3, 4) 和下一时刻的状态。其中动作 (0, 1, 2, 3, 4) 都是动作的编码，它们分别表示前行、左转、右转、繁殖、播种，也就是说每个生命体在每个周期所采用的动作完全是由它们的规则决定的。其中繁殖就是在当前位置诞生一个新的生命体，并且父亲的所有状态都会遗传给后代。

3. 环境规则：如果生命体所在的世界位置有食物，那么就会把这个食物吃掉，吃到食物就可以增加生命体的一定量的能量值。另外，生命体的每一种行动都会对应不同的能量消耗值，而新出生的生命体都会从它的父亲那里得到一部分能量值，如果一个生命体的能量消耗殆尽，或者这个生命体的寿命超过了最大寿命就会死亡。为了限制生命体的繁殖，每个生命都有一个最大繁殖数量。

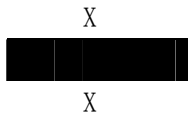
4. 遗传规则：当每次遗传的时候，父亲会把规则表完全遗传给后代，但同时在遗传的过程中会以概率 muteP 发生变异，并且，规则表的长度也会以 lenP 的概率发生变化（既可能增长也可能缩短）。生命体可以通过遗传变异来给自己编程。

在人工生命看来，生命的本质实际上是信息。信息的变化就表现出生命。人工生命的研究和突破说明，信息、算法和计算等概念已经成为理解生命本质的重要概念。人们不仅可以在自然环境中，用人工的方法创造出活的人工生命，甚至还可以在计算机的虚拟环境中，创造出活的数字生物来。这些活的人工生物不仅可以复制自己，而且可以发生变异，甚至可以为某种资源发生生存竞争，从而使自己的后代变得能够更高效地利用这些资源。

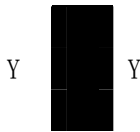
习 题

1. 简述人工生命的基本含义。
2. 试讨论研究人工生命的必要性。
3. 结合人工生命的研究现状，给出一些人工生命程序的例子。
4. 什么是细胞自动机，什么是细胞自动机的变换函数。
5. 结合具体系统解释说明形态形成理论。

6. 每个主体(agent)模型采用可以变化规则表长度的有限自动机模型建模, 设计一种人工生命系统, 可以通过变化不同的食物添加规则研究主体与环境的关系。
7. 设计可以避障的人工鱼。
8. 考虑下图 a 中的状态, 这里只有两个方框(用 x 标出)它们恰好有三个被占用了的邻居。在下一生命周期产生图 b。同样这里只有两个方框(用 y 标出)它们恰好有三个被占用了的邻居。显而易见, 这个社会状态在图 a 和图 b 之间循环, 产生闪光现象邻域集合。请编写程序实现这个生命游戏。



a



b

产生闪光现象邻域集合

9. 随着生物技术的发展, 将会进一步实现遗传学与计算机的融合。一方面, 计算机将进入人体内, 并成为我们身体的一部分; 另一方面, 人工智能的发展很可能意味着, 人类将把自己在智力方面的最高地位转让给会思想的机器。因此, 有人预言计算机将会是下一个生命形式。你同意这种观点吗? 谈谈你的看法。

参考文献

- [1] Agrawal R, T Imieliski, and A Swami, 1993. Mining association rules between sets of items in large database. In Proceedings of ACM SIGMOD International Conference on Management of Data(SIGMOD'93), 207-216, May
- [2] Agre, P. & D. Chapman, 1987. PENGI: An implementation of a theory of activity. In Proc. of the 6th National Conference on AI (AAAI-87), Seattle, WA, USA, 268-272
- [3] Aha, D.A., 1990. Study of Instance-Based Algorithms for Supervised Learning Tasks: Mathematical, Empirical, and Psychological Evaluations. Ph.D. thesis
- [4] Aha, D., Kibler, G. and Albert, M. K., 1991. Instance-Based Learning Algorithms, *Machine Learning*, 6(1), 37-66
- [5] Aha, D., 1997. Lazy Learning. Kluwer Academic Pub.
- [6] Aleksander, I. (editor), 1989. Neural computing architectures, North Oxford Academic.
- [7] Ali, K.M., 1989. Augmenting domain theory for explanation-based generalization, In Proceedings of IWML-89, Cornell University, Ithaca, New York.
- [8] Allen, J.F., 1984. Towards a general theory of action and time, *Artificial Intelligence*, 23: 123-154.
- [9] Amarel, S., 1968. On Representation of Problems of Reasoning about Actions, In D. Michie (editor), *Machine Intelligence 3*, University of Edinburgh Press.
- [10] Amarel, S., 1984. Expert Behavior and Problem Representation, In A. Elithorn and R. Banerji (editors), *Human and Artificial Intelligence*, North-Holland.
- [11] Amarel, S., 1986. Program synthesis as a theory formation task: problem representations and solution methods, In R.S. Michalski, J.G. Carbonell and T.M. Mitchell (editors), *Machine Learning: An Artificial Intelligence Approach*, Vol. II, Morgan Kaufmann.
- [12] Amari S., 1985. Differential Geometrical Methods in Statistics, Springer Lecture Notes in Statistic, 28, Springer
- [13] Amsterdam, J., 1988. Extending the Valiant Learning Model, In Proceedings of ICML-88, San Mateo, CA.
- [14] Amsterdam, J., 1988. Some philosophical problems with formal learning theory, In Proceedings of AAAI-88, Saint Paul, Minnesota.
- [15] Anderson, J.R., 1989. A theory of the origins of human knowledge, *Artificial Intelligence*, 40(1): 313-351.
- [16] Angluin, D. and C.H. Smith, 1983. Inductive Inference: Theory and Methods, *ACM Computing Survey*, 15(3): 237-70.
- [17] Angluin, D., 1986. Learning regular sets from queries and counter-examples, TR-464, New Haven, CT: Yale University, Dept. of Computer Science.
- [18] Angluin, D., 1988. Queries and concept learning, *Machine Learning*, 2(3): 319.
- [13] Ashley, K. D., and Rissland, E. L., 1988. Compare and Contrast: A Test of Expertise, in J.L. Kolodner, editor, *Proceedings of a DARPA Workshop on Case-based Reasoning*, Florida, May.
- [19] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.), 2003. The

Description Logic Handbook. Theory, Implementation and Applications. Cambridge.

- [20] Bakker, R. R., F. Dikker, F. Tempelman and P. M. Wognum, 1993. Diagnosing and solving over-determined constraint satisfaction problems, IJCAI-93, 276-281.
- [21] Banerji, R.B. and T.M. Mitchell, 1980. Description languages and learning algorithms: a paradigm for comparison, International of Policy Analysis and Information Systems, 4(2): 124-40.
- [22] Banerji, R.B., 1985. The logic of learning: a basis for pattern recognition and for improvement of performance, Advances in Computers.
- [23] Banerji, R.B., 1987. A discussion of report by Ehud Shapiro. Comput. Intell., 3, 297-303.
- [24] Bareiss, R., 1988. PROLOS: a unified approach to concept representation, classification and learning, Ph.D. Dissertation, University of Texas at Austin, Dept. of Comp. Sci.
- [25] Barletta, R. and W. Mark, 1988. Explanation-Based Indexing of Cases, In Proceedings of AAAI-88, Saint Paul, Minnesota.
- [26] Barletta, R. and R. Kerber, 1989. Improving explanation-based indexing with empirical learning, In Proceedings of IWML-89, Cornell University, Ithaca, New York.
- [27] Barr, A. and E.A. Feigenbaum, (editors), 1981. The Handbook of Artificial Intelligence, 1, Kaufmann.
- [28] Barr, A. and E.A. Feigenbaum (editors), 1982. The Handbook of Artificial Intelligence, 2, 3, Kaufmann.
- [29] Belew, R.K. and S. Forrest, 1988. Learning and programming in classifier systems, Machine learning, 3(2): 193-223.
- [30] Bellman, R., 1957. Dynamic Programming. Princeton Press.
- [31] Benferhat, S., C. Cayrol, D. Dubois, J. Lang and H. Prade, 1993. Inconsistency management and prioritized syntax-based entailment, IJCAI-93, 640-647.
- [32] Bergadano, F. and A. Giordana, 1988. A knowledge intensive approach to concept induction, In Proceedings of ICML-88, San Mateo, CA.
- [33] Berwick, R.C., 1983. Learning word meanings from examples, In Proceedings of the Eighth IJCAI, Karlsruhe, W. Ger.
- [34] Berwick, R.C., 1986. Domain-independent learning and the subset principle. In R.S. Michalski, J.G. Carbonell and T.M. Mitchell (editors), Machine Learning: An Artificial Intelligence Approach, Morgan Kaufmann.
- [35] Biberman Y. 1994. A Context Similarity Measure. Machine Learning: ECML-94, Springer-Verlag, Berlin Heidelberg
- [36] Blum, L. and M. Blum, 1975. Toward a mathematical theory of inductive inference. Information and Control, 28, 125-55.
- [37] Blumer, A., A. Ehrenfeucht, D. Haussler and A. Warmuth, 1986. Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension, In Proceedings of 18th Annual ACM Symposium on Theory of Computation, Berkeley, CA.
- [38] Bode, J., Zhongzhi Shi, etc. 1995. Neural Networks in New Product Development. Proceedings of CAPE-95, UK.
- [39] Boden, M.A., 1988. Computer Models of Mind. Cambridge University Press.
- [40] Booker, L.B., 1982. Intelligent behavior as an adaptation to the task environment.
- [41] Booker, L.B., 1988. Classifier Systems that Learn Internal World Models. Machine Learning, 3(2): 161-192.
- [42] Booker, L.B., D.E. Goldberg and J.H. Holland, 1989. Classifier Systems and Genetic

Algorithms. Artificial Intelligence, 40(1): 235-282.

- [43] Bradshaw, G.L., P.W. Langley and H.A. Simon, 1980. BACON.4: the discovery of intrinsic properties. In Proceedings of the Canadian Society for Computational Studies of Intelligence, Victoria, B.C.
- [44] Bratman, M. E., Israel, D. J. and Pollack, M. E. Toward an architecture for resource-bounded agents. Technique report CSLI-87-104, Center of Study of Language and Information, SRI and Stanford University, 1987
- [45] Braveman, M.S. and S.J. Russell, 1988. IMEX: overcoming intractability in explanation based learning. In Proceedings of AAAI-88, Saint Paul, Minnesota.
- [46] Braverman, M.S. and S.J. Russell, 1988. Boundaries of operationality. In Proceedings of ICML-88, San Mateo, CA.
- [47] Brazdil, P., 1981. A model for error detection and correction. Dept. of Computer Science, University of Edinburgh.
- [48] Brooks, R.A., 1991. Intelligent without representation. Artificial Intelligence, Vol. 47, 139-159.
- [49] Brooks, R.A., 1991. Intelligence without reasoning. In Proceedings of IJCAI'91, Sydney.
- [50] Buchanan, B.G. and T.M. Mitchell, 1978. Model-directed learning of production rules. In D.A. Waterman and F. Hayes-Roth, (editors), Pattern-Directed Inference Systems, Academic Press.
- [51] Buchanan, B.G. and E.A. Feigenbaum, 1978. Dendral and meta-dendral: their applications dimension. Artificial Intelligence, 11, 5-41.
- [52] Buchanan, B.G., T.M. Mitchell, R.G. Smith and C.R.Jr. Johnson, 1979. Models of learning systems. Dept. of Computer Science, Stanford University.
- [53] Buchanan, B.G., J. Sullivan, T. Cheng and S.H. Clearwater, 1988. Simulation-assisted inductive learning, In Proceedings of AAAI-88, Saint Paul, Minnesota.
- [54] Buggleton, S.H. and W. Buntine, 1988. Towards constructive induction in first-order predicate calculus. Working paper, Turing Institute.
- [55] Bundy, A., 1983. The computer modelling of mathematics reasoning. Academic Press.
- [56] Bunt, Harry, Zhongzhi Shi (Eds.), 1994. International Workshop on Knowledge Engineering and Applications, 1994.
- [57] Buntine, W., 1989. A critique of the Valiant model. In Proceedings of IJCAI-89, Detroit, Michigan, USA.
- [58] Buntine, W., 1991. Classifiers: A theoretical and empirical study. In Proceedings of IJCAI-91, 638-644.
- [59] Burges J C J., 1998. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2):-167
- [60] Burke R and Kass A. 1996. Retrieving Stories for Case-Based Teaching. Case-Based Reasoning (Experiences, lessons, &Future Directions), AAAI/MIT Press, 93-110
- [61] Burstein, M.H., 1986. A model of learning by incremental analogical reasoning and debugging. In Machine Learning: An Artificial Intelligence Approach, Los Altos, Calif.
- [62] Callan, J. P., Fawcett, T. E., and Rissland, E. L., 1991. CABOT: An Adaptive Approach to Case-based Search. in IJCAI-91, Australia.
- [63] Carbonell, J.G., 1981. A computational model of problem solving by analogy, In Proceedings of the Seventh IJCAI, Vancouver, B.C.

- [64] Carbonell, J.G., 1982. Experimental learning in analogical problem solving. In Proceedings of AAAI82, Pittsburgh, Pa.
- [65] Carbonell, J.G., 1983. Derivational analogy and its role in problem solving. In Proceedings of AAAI83, Washington, D.C.
- [66] Carbonell, J.G., R.S. Michalski and T.M. Mitchell, 1983. Machine learning: a historical and methodological analysis, *AI Magazine*,4(3): 69-79.
- [67] Carbonell, J.G., 1983. Learning by analogy: formulating and generalizing plans from past experience. In R.S. Michalski, J.G. Carbonell and T.M. Mitchell (editors), *Machine Learning: An Artificial Intelligence Approach*, Tioga.
- [68] Carbonell, J.G., 1986. Analogy in problem solving. In R.S. Michalski, J.G. Carbonell and T.M. Mitchell (editors), *Machine Learning: An Artificial Intelligence Approach*, Morgan Kaufmann.
- [69] Carbonell, J.G., 1989. Introduction: paradigms for machine learning. *Artificial Intelligence*, 40(1): 1-9.
- [70] Carnap. R., 1950. Logical foundations of probability. The University of Chicago Press.
- [71] Caruana, R.A., L.J. Eshelman and J.D. Schaffer, 1989. Representation and hidden bias II: eliminating defining length bias in genetic search. In Proceedings of IJCAI-89, Detroit, Michigan, USA.
- [72] Catlett, J., 1991. Overpruning large decision trees. In Proceedings of IJCAI-91, 764-769.
- [73] Chang, C.L. and R.C. Lee, 1973. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press.
- [74] Charniak, E. and D. McDermott, 1985. *Introduction to artificial intelligence*. Addison-Wesley Publishing Com.
- [75] Chien, S.A., 1989. Using and refining simplifications: explanation-based learning of plans in intractable domains. In Proceedings of IJCAI-89, Detroit, Michigan, USA.
- [76] Chickering D and Heckerman D. 1996. Efficient approximations for the marginal likelihood of incomplete data given a Bayesian network. Technical Report MSR-TR-96-08, Microsoft Research, Redmond, WA
- [77] Clark, P. and T. Niblett, 1989. The CN2 induction algorithm. *Machine Learning*, 3(4): 261.
- [78] Clements, J., 1982. Analogical reasoning patterns in expert problem solving. In Proceedings of the Fourth Annual Meeting of the Cognitive Science Society, Ann Arbor, Mich.
- [79] Cohen, W.W., 1988. Generalizing number and learning from multiples in explanation based learning. In Proceedings of ICML-88, San Mateo, CA.
- [80] Cook, D.J., 1991. The base selection task in analogical learning. In Proceedings of IJCAI-91, 790-795.
- [81] Cooper, M.C., 1989. An optimal k-consistency algorithm. *Artificial Intelligence*, 41: 89-95.
- [82] Cooper Greg F, and Herskovits E. 1992. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning*, 1992(9): 309-347
- [83] Cui, Z., A.G. Cohn, D.A. Randell, 1992. Qualitative simulation based on a logical formalism of space and time. *AAAI-92*, 679-684.
- [84] Curd, M.V., 1980. The logic of discovery: an analysis of three approaches. In Nickles, T. (editors), *Scientific discovery, logic, and rationality*, D. Reidel Publishing Company.

- [85] DARPA, 1989. Case-based reasoning. In Proceedings of Case-based Reasoning Workshop.
- [86] Danyluk, A.P., 1987. The use of explanations for similarity-based learning. In Proceedings of IJCAI-87, Milan, Italy.
- [87] Darden, L., 1983. Reasoning by analogy in scientific theory construction. In Proceedings of the International Machine Learning Workshop, Allerton House.
- [88] Dasarathy, B.V. 1991. Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques. IEEE Computer Society Press, Los Alamitos, CA.
- [89] Davies, S., 1997. Multidimensional triangulation and interpolation for reinforcement learning. In: Michael C Mozer, Michael I Jordan, Thomas Petsche, eds. Advances in Neural Information Processing Systems 9, NY: MIT Press, 1005-1010.
- [90] Davies, T.R. and S.J. Russell, 1987. A logical approach to reasoning by analogy. In Proceedings of IJCAI-87, Milan, Italy.
- [91] Davis, A.L. and A. Rosenfeld, 1981. Cooperating Process for low-level vision: A survey. Artificial Intelligence, 17:245-263.
- [92] Davis, D., 1987. Constraint propagation with interval labels. Artificial Intelligence, 32: 281-331.
- [93] Davis, L. (editor), 1987. Genetic algorithms and simulated annealing, Pitman.
- [94] Davis, R., 1984. Diagnosis reasoning from structure and behavior, Artificial Intelligence, 24: 347-410.
- [95] Davis, R., 1983. Diagnosis via Causal Reasoning: Paths of Interaction and the Locality Principles, in: Proceedings AAAI-83, Washington, D.C., 88-92.
- [96] de Kleer, J., 1984. How circuits work, Artificial Intelligence, 24: 205-280.
- [97] de Kleer, J., and J. Brown, 1984. A qualitative physics based on confluences, Artificial Intelligence, 24: 205-280.
- [98] de Kleer, J., 1986. An assumption-based TMS, Artificial Intelligence, 28: 127-162.
- [99] de Kleer, J., 1989. A comparison of ATMS and CSP techniques. In Proceedings of IJCAI-89, Menlo Park, 290-296.
- [100] de Kleer, J., 1993. A view on qualitative physics. Artificial Intelligence, 59: 105-114.
- [101] Dechter, R. and J. Pearl, 1987. Network-based heuristics for constraint network. Artificial Intelligence, 34: 1-38.
- [102] Dechter, R. and J. Pearl, 1987. The cycle-cutset method for improving search performance in AI applications. in Proceedings 3rd IEEE on AI Applications, Orlando, FL.
- [103] Dechter, R. and J. Pearl, 1987. Tree clustering for constraint networks, Artificial Intelligence, 28: 342-403.
- [104] Dechter, R., 1989/90. Enhancement for Constraint Processing: Backjumping, Learning, and Cutset Decomposition, Artificial Intelligence, 41: 273-312.
- [105] Decker, K. and V. Lesser, 1993. An approach to analyzing the need for meta-level communication, IJCAI-93, 360-366.
- [106] Decker, K. Environment Centered Analysis and Design of Coordination Mechanisms. PhD thesis, University of Massachusetts, 1995
- [107] Deerwester S, Dumais S T, Furnas G W, Landauer T K, Harshman R. 1990. Indexing by latent semantic analysis, Journal of the American Society for Information Science, 41
- [108] DeJong, G., 1979. Skimming stories in real time: an experiment in integrated

understanding. Dept. of Computer Science, Yale University.

- [109] DeJong, G., 1981. Generalizations based on explanations. In Proceedings of IJCAI-81, Vancouver, B.C.
- [110] DeJong, G., 1982. Automatic schema acquisition in a natural language environment. In Proceedings of AAAI-82, Pittsburgh, Pa.
- [111] DeJong, G., 1983. Acquiring schemata through understanding and generalizing plans. In Proceedings of IJCAI-83, Karlsruhe, W. Ger.
- [112] DeJong, G., 1986. An approach to learning from observation. In Michalski, R.S., J.G. Carbonell and T.M. Mitchell (editors), Machine Learning: An Artificial Intelligence Approach, Morgan Kaufmann.
- [113] DeJong, G. and R. Mooney, 1986. Explanation-based learning: an alternative view, Machine Learning, 1(2): 145.
- [114] DeJong, G., 1988. Some thoughts on the present and future of explanation-based learning. In Proceedings of the ECAI-88.
- [115] De Jong, K., 1988. Learning with genetic algorithms: an overview, Machine Learning, 3(2): 121.
- [116] Delahaye, J.P., 1987. Formal methods in artificial intelligence. North Oxford Academic Pub. Ltd.
- [117] Dershowitz, N., 1986. Programming by analogy. In Michalski, R.S., J.G. Carbonell and T.M. Mitchell (editors), Machine Learning: An Artificial Intelligence Approach, Morgan Kaufmann.
- [118] Diederich, J., 1989. Learning by instruction in connectionist systems. In Proceedings of IWML-89, Cornell University, Ithaca, New York.
- [119] Dietterich, T.G. and R.S. Michalski, 1981. Inductive learning of structural descriptions: evaluation criteria and comparative review of selected methods, Artificial Intelligence, 16(3): 257-94.
- [120] Dietterich, T.G., B. London and K. Clarkson and G. Dromey, 1982. Learning and inductive inference. In Cohen, P.R. and E.A. Feigenbaum (editors), The Handbook of Artificial Intelligence, Kaufmann.
- [121] Dietterich, T.G., 1982. Learning and inductive inference. In Cohen, P.R. and E.A. Feigenbaum (editors), The handbook of artificial intelligence, Pitman.
- [122] Dietterich, T.G. and R.S. Michalski, 1983. A comparative review of selected methods for learning from examples, In Michalski, R.S. and J.G. Carbonell and T.M. Mitchell (editors), Machine Learning: An Artificial Intelligence Approach, Tioga.
- [123] Dietterich, T.G., 1984. Learning about systems that contain state variables, In Proceedings of AAAI-83, Washington, D.C.
- [124] Dietterich, T.G., 1984, Constraint propagation techniques for theory-driven data interpretation, Dept. of Computer Science, Stanford University.
- [125] Dietterich, T.G. and R.S. Michalski, 1986. Learning to predict sequences, In Michalski, R.S. and J.G. Carbonell and T.M. Mitchell (editors), Machine Learning: An Artificial Intelligence Approach, Morgan Kaufmann.
- [126] Dietterich, T.G., 1986. Learning at the knowledge level, Machine Learning, 1(3): 287.
- [127] Dietterich, T.G., 1989. Limitations on inductive learning. In Proceedings of IWML-89, Cornell University, Ithaca, New York.

- [128] Dietzen, S. and F. Pfenning, 1989. Higher-order and model logic as a framework for explanation-based generalization. In Proceedings of IWML-89, Cornell University, Ithaca, New York.
- [129] Domeshek E, Kolodner J and Zimring C. 1994. The Design of a Tool Kit for Case-Based Design Aids, In *Artificial Intelligence in Design*, Norwell, Ma., Kluwer.
- [130] Douglas, S.A. and T.P. Moran, 1983. Learning operator semantics by analogy. In Proceedings of AAAI-83, Washington, D.C.
- [131] Doyle, J., 1979. A truth maintenance system. *Artificial Intelligence*, 12(3): 231-72.
- [132] Durfee, E.H., V.R. Lesser and D.D. Corkill, 1987. Cooperation through communication in a distributed problem solving network, in *Distributed Artificial Intelligence*, Pitman Publishing.
- [133] Ebbinghaus, H-D., 1985. Extended logics: The general framework. in *Model-Theoretic Logic*, Spriger-Verlag.
- [134] Ellman, T., 1988. Approximate theory formation: an explanation-based approach, In Proceedings of AAAI-88, Saint Paul, Minnesota.
- [135] Ellman, T., 1989. Explanation-based learning: programs and perspectives, *ACM Computing Surveys*, 21(2): 163-221.
- [136] Epstein, S.L., 1987. On the discovery of mathematical theorems. In Proceedings of IJCAI-87, Milan, Italy.
- [137] Ernst, G.W. and M.M. Goldstein, 1982. Mechanical discovery of classes of problem-solving strategies, *Journal of the ACM*, 29(1): 1-23.
- [138] Etzioni, O., 1988. Hypothesis filtering: a practical approach to reliable learning. In Proceedings of ICML-88, San Mateo, CA.
- [139] Evance, T.G., 1968. A program for the solution of a class of geometric-analogy intelligence test questions. In M. Minsky (editors), *Semantic Information Processing*, MIT Press.
- [140] Falkenhainer, B., 1985. Proportionality graphs, units analysis, and domain constraints: improving the power and efficiency of the scientific discovery process, In Proceedings of IJCAI85, Los Angeles, Calif.
- [141] Falkenhainer, B., 1987. An examination of the third stage in the analogy process: verification-based analogical learning, In Proceedings of IJCAI-87, Milan, Italy.
- [142] Falkenhainer, B., 1987. Scientific theory formation through analogical inference, In Proceedings of the 4WML, University of California, Irvine.
- [143] Falkenhainer, B.C. and R.S. Michalski, 1986. Integating quantitative and qualitative discovery: the ABACUS system, *Machine Learning*, 1(4): 367.
- [144] Fayyad, U., G. Piatetsky-Shapiro, and P. Smyth, 1996, From data mining to knowledge discovery in databases, *AI Magazine*, Fall, 37-54.
- [145] Fayyad U, Piatetsky-Shapiro, Smyth, and Uthurusamy. 1996a. *Advances in Knowledge Discovery and Data Mining*, MIT Press
- [146] Feigenbaum, E.A. and J. Feldman, 1963. *Computers and Thought*. McGraw-Hill.
- [147] Feldman, J.A. and D.H. Ballard, 1982. Connectionist models and their properties, *Cognitive Science*, 6, 205-254.
- [148] Feldman, J.A., 1982. Dynamic connections in neural networks. *Biol. Cybern.*, 46, 27-39.
- [149] Ferber, J., 1991. Actors and agents as reflective concurrent objects: a mering IV perspective, *SMC-21*: 991.

- [150] Ferguson, I.A. Towards an architecture for adaptive, rational, mobile agents. In Werner and Demazeau(eds.) Decentralised AI 3--Proc. of the 3rd European Workshop on Mod, 1991
- [151] Fikes, S. and P.E. Hart and N.J. Nilsson, 1972. Learning and executing generalized robot plans, Artificial Intelligence, 3(4): 251-88.
- [152] FIPA. FIPA Application Types. <http://www.cselt.stet.it/fipa/yorktown/nyapplications.htm>, 1996
- [153] Fisher, D.H. and P. Langley, 1985. Approaches to conceptual clustering, In Proceedings of IJCAI-85, Los Angeles, Calif.
- [154] Fisher, D.H., 1987. Improving inference through conceptual clustering. In Proceedings of AAAI-87, Seattle, Washington.
- [155] Fisher, D.H., 1987. Knowledge acquisition via incremental conceptual clustering, Machine Learning, 2(2): 139.
- [156] Fisher, D.H., 1987. Knowledge acquisition via incremental conceptual clustering. Machine Learning, 2: 139-172.
- [157] Fisher, D.H. and K.B. McKusick, 1989. An empirical comparison of ID3 and back-propagation, In Proceedings of IJCAI-89, Detroit, Michigan USA.
- [158] Fisher, D.H., 1989. Noise-tolerant conceptual clustering. In Proceedings of IJCAI-89, Detroit, Michigan USA.
- [159] Fitzpatrick, J.M. and J.J. Grefenstette, 1988. Genetic algorithms in noisy environments, Machine Learning, 3(2): 101.
- [160] Flach, P.A., 1987. Second-order inductive learning. In Jantke, K.P., (editors), Analogical and Inductive Inference, Springer-Verlag.
- [161] Forbus, K.D., 1984. Qualitative Process Theory. Artificial Intelligence, Vol. 24, 96-168, 1984.
- [162] Forbus, K.D. and D. Gentner, 1986. Learning physical domains: toward a theoretic framework. In Michalski, R.S. and J.G. Carbonell and T.M. Mitchell (editors), Machine Learning: An Artificial Intelligence, Morgan Kaufmann.
- [163] Forbus, K.D., P. Nielsen and B. Faltings, 1991. Qualitative spatial reasoning: the CLOCK project, Artificial Intelligence, Vol. 51, 417-471.
- [164] Forbus, K.D., 1993. Qualitative Process Theory: twelve years after, Artificial Intelligence, 59, 115-123.
- [165] Forsyth, R., 1984. Machine learning systems. In Proceedings of the Association for Library and Information Management, London.
- [166] Freeman-Benson, B., and A. Borning, 1992, Integrating Constraints with an object-Oriented language, In Proceedings of the 1992 European Conference on Object-Oriented Programming, 268-286.
- [167] Freund Y, Schapire R E. 1995. A decision-theoretic generalization of on-line learning and an application to boosting. Proc. Of the Second European Conference on Computational Learning
- [168] Freuder, E.L., 1978. Synthesizing constraint expression. Communications of the ACM, 21(11): 958-966.
- [169] Freuder, E.L., 1982. A sufficient condition of backtracking-free search. J. ACM, 29(1): 24-32.
- [170] Freuder, E.L., 1988. Backtrack-Free and backtrack bound Search. In search in Artificial Intelligence, L. Kanal and V. Kumar (eds.), 343-369, New York: Springer-Verlag.

- [171] Freuder, E.L., 1989. Partial constraint satisfaction, in Proceedings of IJCAI-89, 278-283.
- [172] Fukushima, K., 1975. Cognitron: a self-organizing multilayered neural network, *Biol. Cybern.*,20: 121-136.
- [173] Gallier, J.H., 1986. *Logic for Computer Science*. Harper and Row.
- [174] Galton, A., 1993. Towards an integrated logic of space, time and motion. IJCAI-93, 1550-1555.
- [175] Genello, R. and F. Mana, 1991. Rigel: an inductive learning system. *Machine Learning*, 6(1): 7-35.
- [176] Gasser, L., C. Bragaza and N. Herman, 1987. MACE: A flexible testbed for distributed AI research, in *Distributed Artificial Intelligence*, Pitman Publishing.
- [177] Genesereth, M.R. and N.J. Nilsson, 1987. *Logicl Foundation of Artificial Intelligence*. Morgan Kaufmann.
- [178] Gennari, J.H., Pat. Langley and Doug Fisher, 1989. Models of incremental concept formation, *Artificial Intelligence*, 40(1): 11-61.
- [179] Gennari, J.H., 1989. Focused concept formation. In *Proceedings of IWML-89*, Cornell University, Ithaca, New York.
- [180] Gentner, D., 1983. Structure mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2): 155-70.
- [181] Gentner, D. and A.L. Stevens, 1983. *Mental Models*. Erlbaum.
- [182] Gentner, D., 1983. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2): 155-170.
- [183] Gerwin, D.G., 1974. Information processing, data inferences, and scientific generalization. *Behav. Sci.*, 19: 314-325.
- [184] Gick, M.L. and K.J. Holyoak, 1980. Analogical problem solving. *Cognitive Psychology*, 12: 306-55.
- [185] Gick, M.L. and K.J. Holyoak, 1983. Schema induction and analogical transfer. *Cognitive Psychology*, 15: 1-38.
- [186] Ginsberg, A., 1988. Theory revision via prior operationalization, In *Proceedings of AAAI-88*, Saint Paul, Minnesota.
- [187] Ginsberg, M. L., 1986. Counterfactuals, *Artificial Intelligence* 30: 35-79.
- [188] Goebel, R., 1987. A sketch of analogy as reasoning with equality hypotheses. In Jantke, K.P. (editors), *Analogical and Inductive Inference*, Springer-Verlag.
- [189] Gold, E.M., 1967. Language identification in the limit. *Information and Control*, 10: 447-74.
- [190] Goldberg, D.E., 1985. Dynamic system control using rule learning and genetic algorithm. in *Proceedings of IJCAI-85*, Los Angeles, Calif.
- [191] Goldberg, D.E. and H. Holland, 1988, *Genetic algorithms and machine learning*. *Machine Learning*, 3(2): 95.
- [192] Goldberg, D.E., 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Com., Inc.
- [193] Golding, A., P.S. Rosenbloom and J.E. Laird, 1987. Learning general search control from outside guidance. In *Proceedings of IJCAI-87*, Milan, Italy.
- [194] Green, C., 1969. Theorem proving by resolution as a basis for question answering systems. In Michie, D. and B. Meltzer (editors), *Machine Intelligence 4*, Edinburgh University

Press.

- [195] Grefenstette, J.J., 1988. Credit assignment in rule discovery systems based on genetic algorithms. *Machine Learning*, 3(2): 225-246.
- [196] Grefenstette, J.J., 1988. Credit assignment in genetic learning systems. in *Proceedings of AAAI-88*, Saint Paul, Minnesota.
- [197] Grefenstette, J.J., 1989. Incremental learning of control strategies with genetic algorithms, In *Proceedings of IWML-89*, Cornell University, Ithaca, New York.
- [198] Greiner, R., 1985. Learning by understanding analogies. STAN-CS-85-1071, Stanford University, Stanford, CA.
- [199] Greiner, R., 1989. Learning by Understanding Analogies. In *Artificial Intelligence*, Vol. 35, No. 1.
- [200] Grossberg, S., 1976. Adaptive pattern classification and universal recoding, I: parallel development and coding of neural feature detectors. *Biol. Cybern.*, 23,121-134.
- [201] Grossberg, S., 1980. How does the brain build a cognitive code? *Psychological Review*, 87, 1-51.
- [202] Grosz, B. and Sarit Kraus. Collaborative plans for complex group actions. *Artificial Intelligence*, 1996, 86(2):269-357
- [203] Gu, J., 1992. Efficient local search for very large-scale satisfiability problems. *Sigart Bulletin*, 3(1): 8-12.
- [204] Guha, R.V. and D.B. Lenat, 1990. Cyc: A midterm report, *AI Magazine*, Fall,32-59.
- [205] Hadzikadic, M. and D.Y.Y. Yun, 1989. Concept formation by incremental conceptual clustering. In *Proceedings of IJCAI-89*, Detroit, Michigan USA.
- [206] Hall, R., 1988. Learning by failing to explain using partial explanations to learn in incomplete or intractable domain. *Machine Learning*, 3(1): 45.
- [207] Hall, R.P., 1989. Computational approaches to analogical reasoning: a comparative analysis. *Artificial intelligence*, 39(1): 39-120.
- [208] Hammond, K.J., 1986. CHEF: A model of case-based planning. in *Proceedings of AAAI-86*, Philadelphia, PA.
- [209] Hammond, K. J., 1990. Explaining and Repairing Plans That Fail, in *Artificial Intelligence* 45.
- [210] Hampson, S.E., 1983. A neural model of adaptive behavior. Dept. of CIS, Univ. of Calif., Irvine.
- [211] Harmelen, F. van and A. Bundy, 1988. Explanation-based generalization = partial evaluation. *Artificial Intelligence*, 36, 401-412.
- [212] Haussler, D., 1987. Learning conjunctive concepts in structural domains, In *Proceedings of AAAI-87*, Seattle, Washington.
- [213] Haussler, D., 1988. New theoretical directions in machine learning. *Machine Learning*, 2(4): 281.
- [214] Haussler, D., 1988. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, 36, 177-221.
- [215] Haussler, D., 1989. Generalizing the PAC model: sample size bounds from metric dimension-based uniform convergence results. in *Proceedings of the Second Annual Workshop on Computational Learning Theory*, Santa Cruz, CA.
- [216] Hayes-Roth, F. and D.J. Mostow, 1981. Machine transformation of advice into a heuristic

- search procedure. In J.R. Anderson (editor), *Cognitive Skills and Their Acquisition*, Erlbaum, Hillsdale, NJ.
- [217] Hayes-Roth, F., 1983. Using proofs and refutations to learn from experience. In R.S. Michalski, J.G. Carbonell and T.M. Mitchell (editors), *Machine Learning: An Artificial Intelligence Approach*, Tigoa.
- [218] Hayes-Roth, F. and D.A. Waterman and D.B. Lenat, 1983. *Building Expert Systems*. Addison Wesley.
- [219] Hebb, D.O., 1949. *The organization of behavior*. Wiley.
- [220] Heckerman D. 1997. Bayesian Networks for Data Mining, *Data Mining and Knowledge Discovery*, 1:79-119
- [221] Hewitt, C. and DeJong, P., 1983. Analyzing the roles of descriptions and actions in open systems, In *AAAI*.
- [222] Hewitt, C., 1991. Open systems semantics for distributed artificial intelligence. *Artificial Intelligence*, 47: 79-106.
- [223] Hinton, G.E., 1989. Connectionist learning procedures. *Artificial Intelligence*, 40(1): 185-234.
- [224] Hirsh, H., 1987. Explanation-based generalization in a logic-programming environment. in *Proceedings of IJCAI-87*, Milan, Italy, 221-227.
- [225] Hirsh, H., 1988. Reasoning about operationality for explanation-based learning. in *Proceedings of ICML-88*, San Mateo, CA.
- [226] Hirsh, H., 1989. Combining empirical and analytical learning with version spaces. in *Proceedings of IWML-89*, Cornell University, Ithaca, New York.
- [227] Hofstadter, D., 1985, *Analogies and roles in human and machine thinking*. In Hofstadter, D. (editors), *Metamagical Themas*, Basic Books.
- [228] Holland, J.H., 1971. Proceeding and processors for schemata. In E.L. Jacks, (editors), *Associative Information Processing*, American Elsevier.
- [229] Holland, J.H., 1975. *Adaptation in Natural and Artificial Systems*, University of Michigan Press.
- [230] Holland, J.H., 1985. Properties of the bucket brigade algorithm. In *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, Pittsburg, PA.
- [231] Holland, J.H., 1986. Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. in Michalski, R.S. and J.G. Carbonell and T.M. Mitchell (editors), *Machine Learning: An Artificial Intelligence Approach*, Morgan Kaufmann.
- [232] Holland, J.H., K.J. Holyoak, R.E. Nisbett and P.R. Thagard, 1986. *Induction: processes of Inference, Learning, and Discovery*, The MIT Press.
- [233] Holyoak, K.J., 1984. Analogical thinking and human intelligence. In Sternberg, R.J. (editors), *Advances in the Psychology of Human Intelligence*, Erlbaum.
- [234] Holyoak, K.J. and P. Thagard, 1989. Analogical mapping by constraint satisfaction. *Cognitive Science*, 13, 295-355.
- [235] Hopfield, J.J., 1982. Neural networks and physical systems with emergent collective computational abilities. In *Proc. Nat. Acad. Sci., USA*.
- [236] Hopfield, J.J. and D.W. Tank, 1985. Neural computation of decisions in optimization problems. *Biol. Cybern*, 52(14): 141-152.
- [237] Hu, Xiaohua, 1995, *Knowledge discovery in database: an attribute-oriented rough set*

approach, Dissertation, Regina.

- [238] Hunt, E.B., J. Marin and P.T. Stone, 1966. Experiments in Induction, Academic Press.
- [239] Iba, G.A., 1979. Learning disjunctive concepts from examples. Master's thesis, MIT.
- [240] Inmon W H. 1992. Building the Data Warehouse. QED Technical Publishing Group
- [241] Inmon, W.H., 1996. The data warehouse and data mining. CACM, No. 11.
- [242] Jennings, N.R. Commitments and Conventions: The foundation of Coordination in Multi-Agent System. The Knowledge Engineering Review, 1993, Vol. 8:3, 233-250
- [243] Johnson-Laird, P.N., 1988. The computer and the mind: an introduction to cognitive science. Harvard University Press.
- [244] Kally, K.T., 1988. Theory discovery and the hypothesis language. in Proceedings of ICML-88, San Mateo, CA.
- [245] Keane, M. T., 1988. Analogical Problem-Solving, Ellis Horwood Limited.
- [246] Kearns, M., Ming Li, L. Pitt and L. G. Valiant, 1987. Recent results on Boolean concept learning, in 4WML.
- [247] Kearns, M. and L. Pitt, 1989. A polynomial-time algorithm for learning k-variable pattern languages from examples. in Proceedings of the Second Annual Workshop on Computational Learning Theory, Santa Cruz, CA.
- [248] Kedar-Cabelli, S., 1987. Formulating concepts according to purpose. in Proceedings of AAAI-87.
- [249] Kedar-Cabelli, S.T., 1985. Purpose-directed analogy. in Proceedings of the Cognitive Science Society, Irvine, Calif.
- [250] Kedar-Cabelli, S.T., 1988. Analogy: from a unified perspective. in D.H. Helman, (editor), Analogical Reasoning: Perspectives of Artificial Intelligence, Cognitive Science, and Philosophy.
- [251] Keller, R.M., 1983. Learning by re-expressing concepts for efficient recognition, in Proceedings of AAAI-83, Washington. D.C.
- [252] Keller, R.M., 1987. Defining operationality for explanation-based learning, in Proceedings of AAAI-87.
- [253] Kerber, M., 1987. Some aspects of analogy in mathematical reasoning, in K.P. Jantke (editor), Analogical and Inductive Inference, Springer-Verlag.
- [254] Kerber, R.G., 1988. Using a generalization hierarchy to learn from examples. in Proceedings of ICML-88, San Mateo, CA.
- [255] Kirsh, D., 1991. Foundations of AI: the big issues. Artificial Intelligence, 47, 3-30.
- [256] Kling, R.E., 1971. A paradigm for reasoning by analogy. Artificial Intelligence, 2(2): 147-78.
- [257] Kodratoff, Y., 1988. Introduction to machine learning. Pitman.
- [258] Kohonen, T., 1984. Self Organization and Associate Memory. Springer-Verlag.
- [259] Kokar, M.M., 1986. Determining arguments of invariant functional descriptions. Machine Learning, 1, 403-422.
- [260] Kolodner, J.L., R.L. Simpson and K. Sycara, 1985. A process model of case-based reasoning in problem solving. in Proceedings of IJCAI-85, Los Angeles, Calif.
- [261] Kolodner, J.L., 1987. Extending problem solver capabilities through case-based inference, in Proceedings of the Fourth International Workshop on Machine Learning, University of California, Irvine.

- [262] Kolodner, J.L., 1988. Retrieving events from a case memory: a parallel implementation. in Proceedings of Case-based Reasoning Workshop.
- [263] Kolodner J L. 1993. *Case-Based Reasoning*. Morgan Kaufmann.
- [264] Koulichev, V. N., 1990. Generalization in Case-based Machine Learning. master thesis, University of Trondheim, Norway, February.
- [265] Koza, J.R., 1989. Hierarchical genetic algorithms operating on populationist learning algorithms. in Proceedings of IJCAI-89, Detroit, Michigan USA.
- [266] Kraus, S. Negotiation and Cooperation in Multi-Agent Environments. Artificial Intelligence Journal, Special Issue on Economic Principles of Multi-Agent Systems, 1997
- [267] Kuipers, B.J., 1984. Common sense reasoning about causality. Artificial Intelligence, Vol. 24, 169-203.
- [268] Kuipers, B.J., 1986. Qualitative simulation. Artificial Intelligence, Vol. 29, 289-338.
- [269] Kuipers, B.J., 1993. Reasoning with qualitative models. Artificial Intelligence, Vol. 59, 125-132.
- [270] Kurtzberg J M. 1987. Feature analysis for symbol recognition by elastic matching, in: *Int'l Business Machines J. of Research and Development* 31:91-9
- [271] V. Kumar, V., 1992. Algorithms for constraint satisfaction problems: a survey. *AI magazine*, 13 (1): 32-44.
- [272] Laird, J., 1988. Recovery from incorrect knowledge in SOAR. in Proceedings of AAAI-88, Saint Paul, Minnesota.
- [273] Laird, J.E, P.S. Rosenbloom and A. Newell, 1986. Chunking in SOAR: The anatomy of a general learning mechanism, *Machine Learning*, 1(1): 11.
- [274] Laird, J.E. and A. Newell, 1983. A universal weak method: summary of results. in Proceedings of IJCAI-83, Karlsruhe, W. Ger.
- [275] Laird, J.E. and P.S. Rosenbloom, 1984. Toward chunking as a general learning mechanism. in Proceedings of AAAI-84, Austin, Tex.
- [276] Laird, J.E., 1984. Universal subgoalting. Dept. of Computer Science, Carnegie-Mellon Univ.
- [277] Laird, P.D., 1988. Learning from good and bad data. Kluwer Academic Publishers.
- [278] Langley, P.W., 1978. BACON. 1: A general discovery system. in Proceedings of the Canadian Society for Computational Studies of Intelligence, Toronto.
- [279] Langley, P.W., 1979. Descriptive discovery processes: experiments in baconian science, Dept. of Psychology, Carnegie-Mellon University.
- [280] Langley, P.W., 1981. Data-driven discovery of physical laws. *Cognitive Science*, 5(1): 31-54.
- [281] Langley, P.W. and H.A. Simon, 1981. The central role of learning in cognition. in Anderson, J.R. (editor), *Cognitive Skill and Their Acquisition*, Erlbaum.
- [282] Langley, P.W., G. Bradshaw and H.A. Simon, 1981. BACON. 5: The discovery of conservation laws. in Proceedings of the IJCAI-81, Vancouver, B.C.
- [283] Langley, P.W., G. Bradshaw and H.A. Simon, 1982. Data-driven and expectation-driven discovery of empirical laws. in Proceedings of the Canadian Society for Computational Studies of Intelligence, Saskatoon, Saskatchewan.
- [284] Langley, P.W., J. Zytkow and H.A. Simon, 1983. Three Facets of Scientific Discovery, in Proceedings of IJCAI-83, Karlsruhe, W. Ger.

- [285] Langley, P.W., H.A. Simon and G.L. Bradshaw, 1983. Rediscovering chemistry with the BACON system. in Michalski, R.S. and J.G. Carbonell and T.M. Mitchell (editors), Machine Learning: An Artificial Intelligence Approach, Tioga.
- [286] Langley, P.W., J.M. Zytkow, H.A. Simon and G.L. Bradshaw, 1986. The search for regularity: Four aspects of scientific discovery. in Michalski, R.S. and J.G. Carbonell and T.M. Mitchell (editors), Machine Learning: An Artificial Intelligence Approach, Morgan Kaufmann.
- [287] Langley, P.W. and R.S. Michalski, 1986. Machine Learning and Discovery. Machine Learning, 1(4): 363.
- [288] Langley, P.W., 1986. On machine learning. Machine Learning, 1(1): 5.
- [289] Langley, P.W., H.A. Simon, G.L. Bradshaw and J.M. Zytkow, 1987. Scientific Discovery. The MIT Press.
- [290] Langley, P.W., 1987. Research papers in machine learning. Machine Learning, 2(3): 195.
- [291] Langley, P.W., 1989. Toward a unified science of machine learning. Machine Learning, 3(4): 253.
- [292] Langley, P.W. and J.M. Zytkow, 1989. Data-driven approaches to empirical discovery. Artificial Intelligence, 40(1): 283-312.
- [293] Langley, P.W., 1989. Unifying themes in empirical and explanation-based learning, In Proceedings of IWML-89, Cornell University, Ithaca, New York.
- [294] Langton, C.G., ed., 1989. Artificial Life. Redwood City, CA: Addison-Wesley.
- [295] Langton, C.G., Taylor, C., Farmer, J.D., Rasmussen, S., eds. 1992. Artificial Life II. Redwood City, CA: Addison-Wesley.
- [296] Langton, C.G., ed. 1994. Artificial Life III. Reading, MA: Addison-Wesley.
- [297] Larkin, J.H., J. McDermott, D.P. Simon and H.A. Simon, 1980. Models of competence in solving physics problems, Cognitive Science, 4, 317-45.
- [298] Leech G, Garside R, Bryant M. 1994. CLAWS: the tagging of the British national corpus. In Proc of 15th Int'l Conf on Computation Linguistics, Kyoto, Japan
- [299] Leler, W., 1988. Constraint programming languages. Their Specification and generation, Addison-Wesley Publishing Company.
- [300] Lenat, D.B., 1976. AM: An artificial intelligence approach to discovery in mathematics as heuristic search. Dept. of Computer Science, Stanford University.
- [301] Lenat, D.B., 1977. Automated theory formation in mathematics. in Proceedings of IJCAI-77, Cambridge, Mass.
- [302] Lenat, D.B., 1983. EURISKO: A program that learns new heuristics and domain concepts: The nature of heuristics III: Program design and results. Artificial Intelligence, 21(1): 61-98.
- [303] Lenat, D.B. and J.S. Brown, 1984. Why AM and eurisko appear to work. Artificial Intelligence, 23(3): 269-94.
- [304] Lenat, D.B., 1988. When will machines learn? in Proceedings of FGCS'88, Tokyo.
- [305] Lenat, D.B. and R.V. Guha, 1990. Building large knowledge-based systems. Addison-Wesley.
- [306] Lenat, D.B., R.V. Guha, K. Pittman, D. Pratt and M. Shepherd, 1990. Cyc: toward programs with common sense, CACM, 33(8).
- [307] Lesser, V.R. A retrospective view of fa/c distributed problem solving. IEEE Transactions on Systems, Man, and Cybernetics, Special Issue on Distributed Artificial Intelligence, 1991
- [308] Levesque, H.J., 1990. All I know: a study in autoepistemic logic. Artificial Intelligence,

42: 263-309.

- [309] Lhomme, O., 1993. Consistency techniques for numeric CSPs. IJCAI-93, 232-238.
- [310] Lesser, V.R., L.D. Erman, 1980. Distributed interpretation: a model and experiment. IEEE Transaction on Computer, C-29: 1144-1163.
- [311] Lesser, V.R., 1991. A retrospective view of FA/C distributed problem solving. IEEE Trans. SMC, SMC-21.
- [312] Liao, L. and Zhongzhi Shi, 1992. Default reasoning in constraint network. Proceedings of the International Workshop on Automated Reasoning, Zhongzhi Shi (ed.), 35-39.
- [313] Liao, L., Zhongzhi Shi, S. Wang, 1994. An approach to the defeasibility of sorted constraint representation, in PRICAI-94.
- [314] Liao, Lejian, Zhongzhi Shi, 1994. Find Preferred Model Using Repair-based Search. PRICAI-94, pp. 80-83.
- [315] Liao, Lejian, Zhongzhi Shi, 1994. Default Reasoning in Sorted Constraint Representation. PRICAI-94, pp. 113-117.
- [316] Liao, Lejian, Zhongzhi Shi, Shijun Wang, 1994. Influence-based Backjumping Combined with Most-Constrained-First and Domain Filtering. DKSME-94, pp. 657-662.
- [317] Liao, Lejian, Zhongzhi Shi, 1995. An Integrated Approach to Constraint Satisfaction. Chinese Journal of Advanced Software Research, Vol. 2, No. 2, pp. 154-160.
- [318] Liao, Lejian, Zhongzhi Shi, 1995. Minimal Model Semantics for Sorted Constraint Representation. J. of Computer Science and Technology, Vol. 10, No. 5, pp. 439-446, 1995.
- [319] Liao, Lejian, Zhongzhi Shi, and D. Zhang, 1995. Minimal Model Semantics for Sorted Constraint Representation, Chinese Journal of Computer science and technology, No. 7.
- [320] Liepins, G.E., M.R. Hilliard, M. Palmer and G. Rangarajan, 1989. Alternatives for classifier system credit assignment. in Proceedings of IJCAI-89, Detroit, Michigan USA.
- [321] Lin, Fangzhen and Reiter, R., 1994. How to progress a database (and why) I: logic foundations. 4th International Conference on Principles of Knowledge Representation and Reasoning.
- [322] Littlestone, N., 1988. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. Machine Learning, 2(4), 285.
- [323] Lloyd, J.W., 1987. Foundations of Logic Programming. Springer-Verlag.
- [324] Luger, George E. 2005 (5th Edition). Artificial Intelligence——Structures and

Strategies for Complex Problem Solving. 史忠植, 张银奎, 赵志崑等译. 人工智能. 机械工业出版社, 2006

- [325] Lynne, K.J., 1988. Competitive reinforcement learning. In Proceedings of ICML-88, San Mateo, CA.
- [326] Mackworth, A.K., 1977. Consistency in networks of relations. Artificial Intelligence, 8(1): 99-118.
- [327] Mahadevan S. and P. Tadepalli, 1988. On the tractability of learning from incomplete theories, in Proceedings of ICML-88, San Mateo, CA.
- [328] Mahadevan, S., 1985. Verification-based learning: A generalized strategy for inferring problem-reduction methods, in Proceedings of IJCAI-85, Los Angeles, Calif.
- [329] Maes, P. The dynamics of action selection. In: Proceedings of IJCAI-89, PG991-997.

Detroit, Michigan, 1989

- [330] Manna, Z. and R. Waldinger, 1985. The Logical basis for computer programming. Addison-Wesley.
- [331] Mark W, Simoudis E, and Hinkle D. 1996. Case-Based Expectations and Results. Case-Based Reasoning (Experiences, lessons, & Future Directions), AAAI/MIT Press, 269-294
- [332] Matheus, C.J. and L.A. Rendell, 1989. Constructive induction on decision trees, in Proceedings of IJCAI-89, Detroit, Michigan USA.
- [333] Mauldin, M.L., 1984. Maintaining diversity in genetic search. In Proceedings of AAAI-84, Austin, Tex.
- [334] McCarthy, J., 1958. Programs with common sense. in Proceedings of the Symposium on the mechanization of Thought Processes.
- [335] McCarthy, J., 1968. Programs with Common Sense. in M. Minsky (editor), Semantic Information Processing, MIT Press.
- [336] McCarthy, J., 1980. Circumscription---a form of non-monotonic reasoning. Artificial Intelligence, 13(1-2): 27-39.
- [337] McCarthy, J., 1986. Applications of circumscription to formalizing commonsense knowledge. Artificial Intelligence, 28: 89-116.
- [338] McCarthy, J. The future of AI-A manifesto. AI Magazine, 26(4): 39, 2005
- [339] McDermott, D. and J. Doyle, 1980. Non-monotonic logic I. Artificial intelligence, 13(1-2): 41-72.
- [340] McClelland, J.L. and D.E. Rumelhart, 1988. Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises, The MIT Press.
- [341] McDermott, D., 1982. Non-monotonic logic II: non-monotonic modal theories. JACM, 29(1): 33-57.
- [342] McDermott, J., 1979. Learning to Use Analogies. in Proceedings of 79, Tokyo.
- [343] McGraw, K.L. and K. Harbison-Briggs, 1989. Knowledge Acquisition: Principles and Guidelines, Prentice Hall.
- [344] Michalski, R.S., 1973. Discovering classification rules using variable-valued logic system VL1. in Proceedings of IJCAI-73, Stanford, Calif.
- [345] Michalski, R.S., 1975. Variable-valued logic and its applications to pattern recognition and machine learning. in D.C. Rine (editor), Computer science and multiple-valued logic theory and applications, North-Holland.
- [346] Michalski, R.S., 1980. Knowledge acquisition through conceptual clustering: A theoretical framework and an algorithm for partitioning data into conjunctive concepts. Policy Analysis and Information Systems, 4(3): 219-44.
- [347] Michalski, R.S. and R.E. Stepp, 1981. An application of AI techniques to structuring objects into an optimal conceptual hierarchy. in Proceedings of IJCAI-81, Vancouver, B.C.
- [348] Michalski, R.S. and R. Stepp and E. Diday, 1981. A recent advance in data analysis: Clustering objects into classes characterized by conjunctive concepts. in Kanal, L. and A. Rosenfeld (editors), Pattern Recognition, North-Holland.
- [349] Michalski, R.S., 1983. A theory and methodology of inductive learning. in R.S. Michalski, J.G. Carbonell and T.M. Mitchell (editors), Machine Learning: An Artificial Intelligence Approach, Tioga.

- [350] Michalski, R.S. and R. Stepp, 1983. Learning from observation: Conceptual clustering. in R.S. Michalski, J.G. Carbonell and T.M. Mitchell (editors), *Machine Learning: An Artificial Intelligence Approach*, Tioga.
- [351] Michalski, R.S., J.G. Carbonell and T.M. Mitchell, 1983. *Machine Learning: An Artificial Intelligence Approach*. Tioga.
- [352] Michalski, R.S., 1984. Inductive learning as rule-guided generalization of symbolic descriptions: A theory and implementation. in A.W. Bierman, G. Guiho and Y. Kodratoff (editors), *Automatic Program Techniques*, Macmillan.
- [353] Michalski, R.S., J.G. Carbonell and T.M. Mitchell, 1986. *Machine Learning: An Artificial Intelligence Approach*, Morgan Kaufmann.
- [354] Michalski, R.S., 1986. Understanding the nature of learning, in Michalski, R.S. and J.G. Carbonell and T.M. Mitchell (editors), *Machine Learning: An Artificial Intelligence Approach*, Morgan Kaufmann.
- [355] Michalski, R.S., S. Amarel, D.B. Lenat, D. Michie and P.H. Winston, 1986. Machine learning: Challenges of the eighties. in R.S. Michalski, J.G. Carbonell and T.M. Mitchell (editors), *Machine Learning: An Artificial Intelligence Approach*, Morgan Kaufmann.
- [356] Michalski, R.S., 1989. *Evolving Research in Machine Learning*. Summer School on Machine Learning.
- [357] Michie, D., 1983. Inductive rule generation in the context of the fifth generation, in *Proceedings of the International Machine Learning Workshop*, Allerton House.
- [358] Mingers, J., 1989. An empirical comparison of selection measure for decision-tree induction, *Machine Learning*, 3(4), 319.
- [359] Minsky, M.L., 1954. *Theory of neural-analog reinforcement systems and its application to the brain-model problem*. Princeton University, Princeton, NJ.
- [360] Minsky, M. and S. Papert, 1969, 1988. *Perceptrons*. MIT Press.
- [361] Minsky, M., 1975. A framework for representing knowledge. in Winston, P.H. (editor), *Psychology of Computer Vision*, McGraw-Hill.
- [362] Minsky, M., 1985. *The Society of mind*. Simon and Schuster Inc.
- [363] Minsky M. 1989. *Semantic Information Processing*. Cambridge, MA.: MIT Press, Hall, R. P.
- [364] Minton, S., A. B. Philips, P. Laird, 1992. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58, 161-205.
- [365] Minton, S., 1984. Constraint-based generalization: Learning game-playing plans from single examples. in *Proceedings of AAAI-84*, Austin, Tex.
- [366] Minton, S., 1985. Selectively generalizing plans for problem-solving. in *Proceedings of IJCAI-85*, Los Angeles, Calif.
- [367] Minton, S. and J.G. Carbonell, 1987. Strategies for learning search control rules: An explanation-based approach. in *Proceedings of IJCAI-87*, Milan, Italy.
- [368] Minton, S., 1988. Quantitative results concerning the utility of explanation-based learning. in *Proceedings of AAAI-88*, Saint Paul, Minnesota.
- [369] Minton, S., 1988. *Learning Search Control Knowledge: An Explanation-Based Approach*. Kluwer Academic Publishers.
- [370] Minton, S., J.G. Carbonell, C.A. Knoblock, D.R. Kuokka, O. Etzioni, Y. Gil, 1989. Explanation-based learning: A problem solving perspective, *Artificial Intelligence*, 40(1): 63-118.

- [371] Mitchell, T.M., 1977. Version spaces: A candidate elimination approach to rule learning. in Proceedings of IJCAI-87, Cambridge, Mass..
- [372] Mitchell, T.M., 1980. The need for biases in learning generalizations,. Technical Report CBM-TR-117, Rutgers University.
- [373] Mitchell, T.M., P.E. Utgoff, B. Nudel and R. Banerji, 1981. Learning problem---solving heuristics through practice. in Proceedings of IJCAI-81, Vancouver, B.C.
- [374] Mitchell, T.M., 1982. Generalization as Search. *Artificial Intelligence*, 18(2): 203-26.
- [375] Mitchell, T.M., P.E. Utgoff and R.B. Banerji, 1983. Learning by experimentation: Acquiring and refining problem---solving heuristics. in R.S. Michalski, J.G. Carbonell and T.M. Mitchell (editors), *Machine Learning: An Artificial Intelligence Approach*, Tioga.
- [376] Mitchell, T.M., 1983. Learning and problem solving. in Proceedings of IJCAI-83, Karlsruhe, W. Ger.
- [377] Mitchell, T.M., 1984. Toward combining empirical and analytic methods for learning heuristics. in A. Elithorn and R. Banerji (editors), *Human and Artificial Intelligence*, North-Holland.
- [378] Mitchell, T.M., S. Mahadevan and L.I. Steinberg, 1985. LEAP: A learning apprentice for VLSI design. in Proceedings of IJCAI-85, Los Angeles, Calif.
- [379] Mitchell, T.M., R.M. Keller and S.T. Kedar-Cabelli, 1986. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1): 47.
- [380] Mo, Chunhuan, Zhongzhi Shi, 1995. Using Genetic Classifier System to Robot Learning. PACES-95, pp. 584-586.
- [381] Mo, Chunhua, Zhongzhi Shi, 1995. Artificial Animate: Sensor and Behavior, ICCADG-95.
- [382] Mohr, R. and Henderson, T.C., 1986. Arc and path consistency revisited. *Artificial Intelligence*, 28: 225-233.
- [383] Montana, D.J. and L. Davis, 1989. Training feedforward neural networks using genetic algorithms. in Proceedings of IJCAI-89, Detroit, Michigan USA.
- [384] Mooney, R. and G. DeJong, 1985. Learning schemata for natural language processing. in Proceedings of IJCAI-85.
- [385] Mooney, R., J. Shavlik, G. Towell and A. Gove, 1989. An experimental comparison of symbolic and connectionist learning algorithms, in Proceedings of IJCAI-89, Detroit, Michigan USA.
- [386] Mooney, R.J., 1988. Generalizing the order of operators in macro-operators. in Proceedings of ICML-88, San Mateo, CA.
- [387] Mooney, R.J., 1989. The effect of rule use on the utility of explanation-based learning. in Proceedings of IJCAI'89, Detroit.
- [388] Moore, A W., 1994. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state spaces. In : Jack D Cowan, Gerald Tesauro, Joshua Alspector, eds. *Advances in Neural Information Processing Systems*, 6: Morgan Kaufmann Publishers, 711-718.
- [389] Moore, R.C., 1985. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25(1): 75-94.
- [390] Moore, R.C. 1993. Autoepistemic logic revisited. *Artificial Intelligence*, 59(1-2): 27-30.
- [391] Mostow, J. and N. Bhatnagar, 1987. Failsafe---a floor planner that uses EBG to learn from

its failures. in Proceedings of IJCAI-87, Milan, Italy.

[392] Mostow, J., 1989. Design by derivational analogy: Issues in the automated replay of design plans. *Artificial Intelligence*, 40(1): 119-184.

[393] Muggleton, S.H. and W. Buntine, 1988. Machine invention of first-order predicates by inverting resolution. in Proceedings of IWML.

[394] Muggleton, S.H. and W. Buntine, 1988. Towards constructive induction in first-order predicate calculus. Turing Institute.

[395] Muhlenbein, H. and J. Kindermann, 1989. The dynamics of evolution and learning --towards genetic neural networks. in R. Pfeifer and Z. Schreter and F. Fogelman-Soulie and L. Steels (editors), *Connectionism in Perspective*, North-Holland.

[396] Muller, B. and J. Reinhardt, 1990. *Neural networks--an introduction*. Springer--Verlag.

[397] Muller, J.P. and Pischel, M. Modelling interacting agents in dynamic environments, In: *Proceedings of the 11th European Conference on Artificial Intelligence (ECAI-94)*, 1994, 709-713

[398] Nadel, B.A., 1989. Constraint Satisfaction algorithms, *Computational Intelligence*, 5(4): 188-224.

[399] Nadel, B., 1990. Tree search and arc consistency in constraint satisfaction algorithms. In *Search in Artificial Intelligence*, eds. L. Kanal and V. Kumar, 287-342. New York, Springer-Verlag.

[400] Natarajan, B.K. and P. Tadepalli, 1988. Two new frameworks for learning. in *Proceedings of ICML-88*, San Mateo, CA.

[401] Newell, A. and H.A. Simon, 1972. *Human Problem Solving*. Prentice-Hall.

[402] Newell, A. and P. Rosenbloom, 1981. Mechanisms of skill acquisition and the law of practice. In Anderson, J.R., (editor), *Cognitive Skills and Their Acquisition*, Erlbaum.

[403] Nickles, T., 1980. *Scientific discovery, logic, and rationality*, D. Reidel Publishing Company.

[404] Nigam K, McCallum A, Thrun S and Mitchell T. 1998. Learning to Classify Text from Labeled and Unlabeled Documents. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 792-799

[405] Nilsson, N.J., 1980. *Principles of Artificial Intelligence*. Tioga.

[406] Norton, S.W., 1989. Generating better decision trees. in *Proceedings of IJCAI-89*, Detroit, Michigan USA.

[407] O'Rorke, P., 1984. Generalization for explanation-based schema acquisition. in *Proceedings of AAAI-84*, Austin, Tex.

[408] Osawa, E.I. A Schema for Agent Collaboration in Open Multiagent Environments. *IJCAI-93*, 1993, 352-359

[409] Padgham, L. and T. Zhang, 1993. A terminological Logic with defaults: A definition and an application. *IJCAI-93*, 662-668.

[410] Pagallo, G., 1989. Learning DNF by decision trees. in *Proceedings of IJCAI-89*, Detroit, Michigan USA.

[411] Pagallo, G. and D. Haussler, 1989. Two algorithms that learn DNF by discovering relevant features. in *Proceedings of IWML-89*, Cornell University, Ithaca, New York.

[412] Paredis, J., 1993. Genetic state-space search for constrained optimization problems, *IJCAI-93*, 952-959.

[413] Pawlak, Z., 1982. Rough sets. *International Journal of Information and Computer Science*, 11(5): 341-356.

- [414] Pawlak, Z., 1991. Rough Sets: Theoretical Aspects of Reasoning About Data. Kluwer Academic Publishers.
- [415] Pazzani, M.J., 1988. Integrated learning with incorrect and incomplete theories. in Proceedings of ICML-88, San Mateo, CA.
- [416] Pazzani, M.J., 1989. Explanation-based learning with weak domain theories. in Proceedings of IWML-89, Cornell University, Ithaca, New York.
- [417] Peters, J.F., Pawlak, Z. and Skowron, A. A rough set approach to measuring information granules, Proceedings of COMPSAC 2002, 1135-1139, 2002.
- [418] Pettit, E. and K.M. Swigger, 1983. An analysis of genetic-based pattern tracking and cognitive-based component tracking models of adaptation. in Proceedings of AAAI-83, Washington, D.C.
- [419] Pitt, L., 1987. Inductive reference, DFAs, and computational complexity. in Jantke, K.P. (editor), Analogical and Inductive Inference, Springer-Verlag.
- [420] Poetschke, D., 1987. Analogical reasoning for second generation expert systems. in Jantke, K.P. (editor), Analogical and Inductive Inference, Springer-Verlag.
- [421] Politakis, P. and S.M. Weiss, 1984. Using empirical analysis to refine expert system knowledge bases. Artificial Intelligence, 22(1): 23-48.
- [422] Popper, K., 1961. The Logic of Scientific Discovery. Science Editions.
- [423] Popper, K., 1968. The Logic of Scientific Discovery. Harper and Row.
- [424] Porat, S., 1991. Learning automata from ordered examples. Machine Learning, 7(2-3): 109-138.
- [425] Porter, B. and D. Kibler, 1984. Learning operator transformations. in Proceedings of AAAI-84, Austin, Tex.
- [426] Porter, B., 1984. Learning problem solving. Department of Computer and Information Science, University of California, Irvine.
- [427] Porter, B.W. and D.F. Kibler, 1986. Experimental goal regression: A method for learning problem-solving heuristics. Machine Learning, 1(3), 249.
- [428] Prieditis, A.E. and J. Mostow, 1987. PROLEARN: Towards a prolog interpreter that learns. in Proceedings of AAAI-87.
- [429] Prosser, P., 1991. Hybrid Algorithms for the constraint satisfaction problem. Research report, AISL-46-91, Computer Science Dept., Univ. of Strathclyde.
- [430] Prosser, P., 1993. Domain filtering can degrade intelligent backtracking search. In Proceedings IJCAI-93, 262-267.
- [431] Prosser, P., 1993. BM + BJ= BMJ, Proceedings of CAIA-93, 257-262.
- [432] Purdom, P., 1983. Search rearrangement backtracking and polynomial average time. Artificial Intelligence, 21: 117-133.
- [433] Quinlan, J.R., 1979. Discovering rules from large collections of examples: A case study. in Michie, D. (editor), Expert Systems in the Micro Electronic Age, Edinburgh University Press.
- [434] Quinlan, J.R., 1983. Learning efficient classification procedures and their application to chess end-games. in R.S. Michalski, J.G. Carbonell and T.M. Mitchell (editors), Machine Learning: An Artificial Intelligence Approach, Tioga.
- [435] Quinlan, J.R., 1986. Induction of decision trees. Machine Learning, 1(1): 81.
- [436] Quinlan, J.R., 1986. The effect of noise on concept learning. in R.S. Michalski, J.G. Carbonell and T.M. Mitchell (editors), Machine Learning: An Artificial Intelligence Approach,

Morgan Kaufmann.

- [437] Quinlan, J.R., 1987. Generating production rules from decision trees. in Proceedings of IJCAI-87, Milan, Italy.
- [438] Quinlan, J.R., 1988. An empirical comparison of genetic and decision-tree classifiers. In Proceedings of ICML-88, San Mateo, CA.
- [439] Rajamoney, S.A., 1988. Experimentation-based theory revision. in Proceedings of the AAAI Spring Symposium on EBL, Menlo Park, Calif.
- [440] Randall, D.A., Z. Cui and A.G. Cohn, 1992. A spatial logic based on regions and connection. ICKRR'92, 165-175.
- [441] Rao A.S. and Georgeff M. P. An Abstract Architecture for Rational Agents. In Proc. of the 3rd International Conference on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann, 1992
- [442] Redmond, M., 1989. Combining case-based reasoning, explanation-based learning, and learning from instruction. in Proceedings of IWML-89, Cornell University, Ithaca, New York.
- [443] Reinke, L.P., 1988. Criteria for polynomial-time(conceptual) clustering. Machine Learning, 2(4): 371.
- [444] Reiter, R., 1978. On closed world data bases. in H. Gallaire and J. Minker, (editors), Logic and Data Bases, Plenum Press.
- [445] Reiter, R., 1980. A logic for default reasoning. Artificial Intelligence, 13(1-2): 81-132.
- [446] Reiter, R. and G. Criscuolo, 1983. Some representational issues in default reasoning. International Journal of Computers and Mathematics, 9(1): 15-27.
- [447] Reiter, R. and J. de Kleer, 1987. Foundation of assumption based truth maintenance systems: Preliminary report. in Proceedings AAAI-87, Seattle WA.
- [448] Richie, G.D. and F.K. Hanna, 1984. AM: A case study in AI methodology. Artificial Intelligence, 23(3): 249-68.
- [449] Rieger, C., and Grinberg, M., 1977. The declarative representation and procedural simulation of causality in physical mechanisms, IJCAI-77, 250-256.
- [450] Riesbeck, C.K., and Schank, R.C., 1989. Inside Case-based Reasoning. Lawrence Erlbaum Associates, Inc., Publishers, Hillsdale, New Jersey.
- [451] Robertson, G.G., 1988. Population size in classifier systems. in Proceedings of ICML-88, San Mateo, CA.
- [452] Robertson, G.G. and R.L. Riolo, 1988. A tale of two classifier systems. Machine Learning, 3(2): 139.
- [453] Robinson, J.A., 1965, A machine-oriented logic based on the resolution principle, JACM, 12(1), 23-41.
- [454] Rose, D. and P. Langley, 1986, Chemical discovery as belief revision, Machine Learning, 1(4), 423.
- [455] Rose, D. and P. Langley, 1988, A hill-climbing approach to machine discovery, in Proceedings of ICML-88, San Mateo, CA.
- [456] Rosenblatt, F., 1962, Principles of Neurodynamics, Spartan Books.
- [457] Rosenbloom, P.S. and A. Newell, 1982, Learning by chunking: summary of a task and a model, in Proceedings of AAAI-82, Pittsburgh, Pa.
- [458] Rosenbloom, P.S., 1983, The chunking of goal hierarchies: A model of practice and stimulus-response compatibility, Department of Psychology, Carnegie-Mellon University.

- [459] Rosenbloom, P.S. and A. Newell, 1986, The chunking of goal hierarchies, in R.S. Michalski, J.G. Carbonell and T.M. Mitchell (editors), *Machine Learning: An Artificial Intelligence Approach*, Morgan Kaufmann.
- [460] Rosenbloom, P.S. and J.E. Laird, 1986, Mapping explanation-based generalization onto SOAR, in *Proceedings of the AAAI-86*, Los Altos, Calif.
- [461] Rosenbloom, P.S., J.E. Laird and A. Newell, 1987, Knowledge Level Learning in SOAR, in *Proceedings of AAAI-87*.
- [462] Rosenschein, J.S. (1986), *Rational Interaction Cooperation Among Intelligent Agents*, Ph.D. Disertation, Stanfont University.
- [463] Roy, S. and J. Mostow, 1988, Parsing to learn fine grained rules, in *Proceedings of AAAI-88*, Saint Paul, Minnesota.
- [464] Rumelhart, D.E. and A.A. Abrahamson, 1973, A model for analogical reasoning, *Cognitive Psychology*, 5, 1-28.
- [465] Rumelhart, D.E. and D.E. Norman, 1981, Analogical processes in learning, in Anderson, J.R., (editor), *Cognitive Skills and Their Acquisition*, Erlbaum.
- [466] Rumelhart, D.E. and D. Zipser, 1985, Competitive Learning, *Cognitive Science*, 9, 75-112.
- [467] Rumelhart, D.E. and J.L. McClelland, 1986, *Parallel distributed processing*, The MIT Press.
- [468] Rumelhart, D.E., G.E. Hinton and R.J. Williams, 1986, Learning internal representations by error propagation, in Rumelhart, D.E. and J.L. McClelland and the PDP Res. Group (editors), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, I: Foundations*, MIT Press.
- [469] Russell, S.J., P. Norvig, 2003. *Artificial Intelligence——A Modern Approach* (2nd Edition). 姜哲等译. *人工智能——一种现代方法* (第二版). 人民邮电出版社, 2004
- [470] Russell, S.J. and B.N. Grosz, 1987, A declarative approach to bias in concept learning, in *Proceedings of AAAI-87*.
- [471] Salzberg, S., 1983, Generating hypotheses to explain prediction failures, in *Proceedings of AAAI-83*, Washington, D.C.
- [472] Salzberg, S. and D.J. Atkinson, 1984, Learning by building causal explanations, in *Proceedings of the Sixth European Conference on AI*, Pisa, Italy.
- [473] Salzberg, S., 1985, Heuristics for inductive learning, in *Proceedings of IJCAI-85*, Los Angeles, Calif.
- [474] Sammut, C., 1981, Learning concepts by performing experiments, Department of Computer Science, University of New South Wales.
- [475] Sammut, C., 1981, Concept learning by experiment, in *Proceedings of IJCAI-81*, Vancouver, B.C.
- [476] Sammut, C. and R.B. Banerji, 1986, Learning concepts by asking questions, in R.S. Michalski, J.G. Carbonell and T.M. Mitchell (editors), *Machine Learning: An Artificial Intelligence Approach*, Morgan Kaufmann.
- [477] Samuel, A.L., 1963, Some studies in machine learning using the game of checkers, in E.A. Feigenbaum and J. Feldman (editors), *Computers and Thought*, McGraw-Hill.
- [478] Shanahan, Murray, 1997. *Solving the Frame Problem*, The MIT Press.
- [479] Sarrett, W.E. and M.J. Pazzani, 1989, One-sided algorithms for integrating empirical and explanation-based learning, in *Proceedings of IWML-89*, Cornell University, Ithaca, New York.

- [480] Schaffer, J.D. and J.J. Grefenstette, 1985, Multi-objective learning via genetic algorithms, in Proceedings of IJCAI-85, Los Angeles, Calif.
- [481] Schank, R.C., Conceptual dependency: a theory of natural language understanding, Cognitive Psychology, Vol. 3, No. 4, 1972.
- [482] Schank, R.C., Dynamic Memory, Cambridge University Press, Cambridge, 1982.
- [483] Schank, R.C., Looking at learning, in Proceedings of the Fifth European Conference on AI, Paris, 1982.
- [484] Schank, R.C. and David B.L., 1989, Creativity and learning in a case-based explainer, Artificial Intelligence, 40(1), 353-385.
- [485] Schapire, R.E., 1989, The strength of weak learnability, in Proceedings of the Second Annual Workshop on Computational Learning Theory, Santa Cruz, CA.
- [486] Schlimmer, J.C. and D. Fisher, 1986, A case study of incremental concept induction, in Proceedings of AAAI-86.
- [487] Schlimmer, J.C. and R.H. Granger, Jr, 1987, Incremental learning from noisy data, Machine Learning, 1(3), 317.
- [488] Schreiber, Guus. Knowledge Engineering and Management. 史忠植, 梁永全, 吴斌等译. 知识工程和知识管理. 机械工业出版社, 2003.
- [489] Segre, A.M., 1987, On the operationality/generality trade-off in explanation-based learning, in Proceedings of IJCAI-87, Milan, Italy.
- [490] Segre, A.M., 1988, Operationality and real world plans, in Proceedings of AAAI-88, Menlo Park, Calif.
- [491] Shafer G. 1976. A Mathematical Theory of Evidence. Princeton, NJ: Princeton University Press
- [492] Shapiro, E.Y., 1981, An algorithm that infers theories from facts, in Proceedings of the IJCAI-81, Vancouver, B.C.
- [493] Shapiro, E.Y., 1981, Inductive inference of theories from facts, Research Report 192, Department of Computer Science, Yale University, New Haven, CT.
- [494] Shapiro, E.Y., 1983, Algorithmic Program Debugging, MIT Press.
- [495] Shavlik, J.W. and G.F. DeJong, 1987, An explanation-based approach to generalizing number, in Proceedings of IJCAI-87, Milan, Italy.
- [496] Shavlik, J.W. and G.F. DeJong, 1987, BAGGER: An EBL system that extends and generalizes explanations, in Proceedings of AAAI-87.
- [497] Shavlik, J.W. and G.G. Towell, 1989, Combining explanation-based learning and artificial neural networks, in Proceedings of IWML-89, Cornell University, Ithaca, New York.
- [498] Shi, Zhongzhi, Intelligent Scheduling Architecture in KSS, The Second International Conference on Computers and Applications, Beijing, 6, 1987.
- [499] Shi, Zhongzhi, On Knowledge Base System Architecture, Proceedings of Knowledge-Based Systems and Models of Logical Reasoning, Cairo, 1988.
- [500] Shi, Zhongzhi, Distributed Artificial Intelligence, Proceedings on the Future of Research in AI, 1989.
- [501] Shi, Zhongzhi, Design and Implementation of FORMS, Proceedings of International Conference on Computer and Applications, Beijing, 1984.
- [502] Shi, Zhongzhi, etc. (eds.), New Generation Computer Systems, International Academic Publishers, 1989, 4.

- [503] Shi, Zhongzhi and J.C. Bioch, Computational Learning Theories, in Proceedings of BENELEARN'90, Katholieke Universiteit Leuven, Belgium, 1990.
- [504] Shi, Zhongzhi and J. Han, Attribute Theory in Learning System, Future Generation Computer Systems, North Holland Publishers, No.6, 1990.
- [505] Shi, Zhongzhi and J. Wu, H. Sun, J. Xu, OKBMS, An Object-Oriented Knowledge Base Management System, International Conference on TAI, Washington D.C. 11, 1990.
- [506] Shi, Zhongzhi, Principles of Machine Learning, International Academic Publishers, 1992.
- [507] Shi, Zhongzhi, Hierarchical Model of Human Mind, PRICAI-92, Seoul, 1992.
- [508] Shi, Zhongzhi, etc., DEIDS: An Integrated Environment for Intelligent Decision Systems, PRICAI-94, 1994.
- [509] Shi, Zhongzhi (Ed.), Proceedings of Pacific Rim International Conference on Artificial Intelligence, International Academic Publishers, 1994.
- [510] Shi, Zhongzhi, etc., MAPE: Multi-Agent Processing Environment, PRICAI-94, 1994.
- [511] Shi, Zhongzhi, Artificial Thought and Intelligent Systems, AI Summer School'94, 1994.
- [512] Shi, Zhongzhi, Tao Wang, Wenjie Wang, Qijia Tian, Meng Ye, 1995. A Flexible Architecture for Multi-Agent System, PACES-95.
- [513] Shi, Zhongzhi, Qijia Tian, Wenjie Wang, Tao Wang, Epistemic Reasoning About Knowledge and Belief Based on Dependence Relation, Advanced Software Research, Vol. 3, No. 2, 1996.
- [514] Shi, Zhongzhi, Jun Wang, Applying Case-Based Reasoning to Engine Oil Design, AI in Engineering, 11 (1997), 167-172.
- [515] Shi, Zhongzhi, Mingkai Dong, Yuncheng Jiang, Haijun Zhang, 2005. A Logic Foundation for the Semantic Web. Science in China, Series F Information Sciences, 48(2): 161-178.
- [516] Shivakumar Vaithyanathan, Byron Dom. 1998. Model-Based Hierarchical Clustering . PRICAI Workshop on Text and Web Mining
- [517] Shoham, Y. (1992), On the synthesis of useful social laws for artificial agent societies, AAAI-92.
- [518] Shoham, Y. (1993), Agent-oriented programming. Artificial Intelligence, 60: 51-92.
- [519] Shoham, Y., Moshe Tennenholtz. On social laws for artificial agent societies: off-line design. Artificial Intelligence, 1995, 73:231-252
- [520] Siekmann, J. and P. Szabo, 1982, Universal Unification, Springer-Verlag.
- [521] Silver, B., 1986, Precondition analysis: learning control information, in R.S. Michalski, J.G. Carbonell and T.M. Mitchell (editors), Machine Learning: An Artificial Intelligence Approach, Morgan Kaufmann.
- [522] Simon, H.A. and G. Lea, 1974, Problem solving and rule induction: A unified view, in Gregg, L.W. (editor), Knowledge and Cognition, Erlbaum.
- [523] Simon, H.A., 1982, The Sciences of the Artificial, MIT Press.
- [524] Simon, H.A., 1983, Why should machines learning?, in R.S. Michalski, J.G. Carbonell and T.M. Mitchell (editors), Machine Learning: An Artificial Intelligence Approach, Tioga.
- [525] Simpson, R.L., 1985, A computer model of case-based reasoning in problem solving, School of ICS, Georgia inst. of Tech., Atlanta, GA.
- [526] Singh, S., T. Jaakkola, M.I. Jordan, 1995.. Reinforcement learning with soft state aggregation. In: G Tesauro, D Touretzky, eds. Advances in Neural Information Processing Systems, 7. Morgan Kaufmann: MIT Press, 361-368.

- [527] Sleeman, D.H., P. Langley and T.M. Mitchell, 1982, Learning from solution paths: An approach to the credit assignment problem, *AI Magazine*, 3(2), 48-52.
- [528] Smith, R., 1980, A learning system based on genetic algorithms, Department of Computer Science, University of Pittsburgh.
- [529] Smith, R.G. (1980), The contract-net protocol: high-level communication and control in a distributed problem solver, *IEEE Transaction on Computers*, C-29(12): 1104-1113.
- [530] Soloway, E.M., 1978, Learning = interpretation + generalization: A case study in knowledge-directed learning, Department of CIS, University of Massachusetts, Amherst.
- [531] Steier, D., 1987, CYPRESS-Soar: A case study in search and learning in algorithm design, in *Proceedings of IJCAI-87*, Milan, Italy.
- [532] Stepp, R.E. (1984), Conjunctive conceptual clustering: A methodology and experimentation, Department of Computer Science, University of Illinois, Urbana.
- [533] Stepp, R.E. and R.S. Michalski (1986), Conceptual clustering: inventing goal-oriented classifications of structured objects, in R.S. Michalski, J.G. Carbonell and T.M. Mitchell (editors), *Machine Learning: An Artificial Intelligence Approach*, Morgan Kaufmann.
- [534] Stepp, R.E., 1987, Concepts in conceptual clustering, In *Proceedings of IJCAI-87*, Milan, Italy.
- [535] Sewell W and Shah V. 1968. Social class, parental encouragement, and educational aspirations. *American Journal of Sociology*, 73:559--572.
- [536] Stone, P., Veloso, M. Task decomposition, dynamic role assignment, and low-bandwidth communication for real time strategic teamwork. *Artificial Intelligence*, 1999, 110(2): 241-273
- [537] Sutton, R. S., 1996. Generalization in reinforcement learning: successful examples using sparse coarse coding. In: D. Touretzky, M. Mozer, M. Hasselmo, eds. *Advances in Neural Information Processing Systems*, 8, NY: MIT Press, 1038-1044.
- [538] Sycara, K., K. Decker, A. Pannu, M. Williamson, D. Zeng. Distributed Artificial Agents. URL: <http://www.cs.cmu.edu/~softagents/>, 1996
- [539] Tadepalli, P., 1989, Lazy explanation-based learning: A solution to the intractable theory problem, in *Proceedings of IJCAI-89*, Detroit, Michigan USA.
- [540] Tadepalli, P., 1991, A formalization of explanation-based macro-operator learning, in *Proceedings of IJCAI-91*, 616-622.
- [541] Tambe, M. and A. Newell, 1988, Some Chunks Are Expensive, in *Proceedings of ICML-88*, San Mateo, CA.
- [542] Tambe, M. and P. Rosenbloom, 1989, Eliminating expensive chunks by restricting expressiveness, in *Proceedings of IJCAI-89*, Detroit, Michigan USA.
- [543] Tesauro, G.J., 1992. Practical Issues in Temporal Difference Learning, *Machine Learning*, Vol. 8, pp. 257-277.
- [544] Thagard, P. and D.M. Cohen and K.J. Holyoak, 1989, Chemical analogies: Two kinds of explanation, in *Proceedings of IJCAI-89*, Detroit, Michigan USA.
- [545] Thagard, P., Holyoak, K. J., Nelson, G. and Gochfeld, D. (1990), Analogy Retrieval by Constraint Satisfaction, in *Artificial Intelligence*, 46.
- [546] Thornton, C.J., 1987, Analogical inference as generalized inductive inference, in Jantke, K.P. (editors), *Analogical and Inductive Inference*, Springer-Verlag.
- [547] Tian, Qijia, Zhongzhi Shi, Wenjie Wang, Tao Wang, An Approach to Autoepistemic Logic, *ICYCS-95*, 1995.

- [548] Tian, Qijia, Zhongzhi Shi, Wenjie Wang, Tao Wang, An Approach to Autoepistemic Logic, ICYCS-95, 1995.
- [549] Tian, Qijia, Zhongzhi Shi, A Model-Theoretical Approach to Action and Progression, SMC-96, 1996.
- [550] Tian, Qijia, Zhongzhi Shi, Wenjie Wang, Tao Wang, An Agent-Oriented Semantics for Multimodal Logic, Chinese Journal of Electronics (to be appeared).
- [551] Tsitsiklis, John N., 1994. Asynchronous stochastic approximation and Q-learning. Machine Learning, 16(3):185-202.
- [552] Tu, Xiaoyuan, 1996. Artificial Animals for Computer Animation. Ph.D. Dissertation, University of Toronto
- [553] Utgoff, P.E. and T.M. Mitchell, 1982, Acquisition of appropriate bias for inductive concept learning, in Proceedings of AAAI-82, Pittsburgh, Pa.
- [554] Utgoff, P.E., 1983, Adjusting bias in concept learning, in Proceedings of IJCAI-83, Karlsruhe, W. Ger.
- [555] Utgoff, P.E., 1984, Shift of bias for inductive concept learning, Department of Computer Science, Rutgers University.
- [556] Utgoff, P.E., 1986, Machine Learning of Inductive Bias, Kluwer Academic Publishers.
- [557] Utgoff, P.E., 1986, Shift of bias for inductive concept learning, in R.S. Michalski, J.G. Carbonell and T.M. Mitchell (editors), Machine Learning: An Artificial Intelligence Approach, Morgan Kaufmann.
- [558] Utgoff, P.E., 1988, Perceptron trees: A case study in hybrid concept representations, in Proceedings of AAAI-88, Saint Paul, Minnesota.
- [559] Utgoff, P.E., 1988, ID5: An incremental ID3, in Proceedings of ICML-88, San Mateo, CA.
- [560] Valiant, L.G., 1984, A theory of the learnable, Communications of the ACM, 27(11), 1134-42.
- [561] Valiant, L.G., 1985, Learning disjunction of conjunction, in Proceedings of IJCAI-85, Los Angeles, Calif.
- [562] VanLehn, K. and W. Ball, 1987, A version space approach to learning context-free grammars, Machine Learning, 2(1), 39.
- [563] Vapnik V N. 1995. The Nature of Statistical Learning Theory. Springer-Verlag, New York
- [564] Vapnik V N. 1998. Statistical Learning Theory. Wiley-Interscience Publication, John Wiley & Sons, Inc
- [565] Vapnik V, Golowich S, Smola A. 1997. Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing, in Advances in Neural Information Processing Systems 9
- [566] Voisin J, Devijver P A. 1987. An application of the Multiedit-Condensing technique to the reference selection problem in a print recognition system. Pattern Recognition 5:465 - 474
- [567] Waddington, C.H., 1974. A catastrophic theory of evolution, Annals of the New York Academy of Science, 231, 32-42.
- [568] Waltz, D., 1975. Understanding line drawings of scenes with shadows. In The Psychology of Computer Vision, P.H. Winston (ed.), 19-91, McGraw Hill.
- [569] Warmuth, M.K., 1987, Towards representation independence in PAC learning, In K.P. Jantke, (editor), Analogical and Inductive Inference, Springer-Verlag.

- [570] Watkins, C., Peter Dayan, 1989. Q-learning. *Machine Learning*, 8, 279-292.
- [571] Weiss, S.M., Kulikowski, C.A. 1991. *Computer systems that learn*. Morgan Kaufmann Publishers, Inc., San Mateo, California.
- [572] Werbos, P.J., 1974, Beyond regression: New tools for prediction and analysis in the behavioral sciences, Harvard University, Cambridge, MA.
- [573] Widmer, G., 1989, A tight integration of deductive and inductive learning, in *Proceedings of IWML-89*, Cornell University, Ithaca, New York.
- [574] Wilkins, D.C., 1988, Knowledge base refinement using apprenticeship learning techniques, in *Proceedings of the AAAI-88*, Los Altos, Calif.
- [575] Williams, R.S., 1988, Learning to program by examining and modifying cases, in *Proceedings of ICML-88*, San Mateo, CA.
- [576] Wilson D. 1972. Asymptotic Properties of Nearest Neighbor Rules using Edited Data. *Institute of Electrical and Electronic Engineers Transactions on Systems, Man and Cybernetics* 2:408 - 421.
- [577] Wilson, G.V. and G.S. Pawley, 1988, On the stability of the traveling salesman problem algorithm of Hopfield and Tank, *Biological Cybernetics*, 58, 62-70.
- [578] Wilson, S.W., 1987. Classifier systems and the animat problem, *Machine Learning*, 2(3), 199.
- [579] Wilson, S.W., 1987. Hierarchical credit allocation in a classifier system, in *Proceedings of IJCAI-87*, Milan, Italy.
- [580] Winston, P.H., 1975. Learning structural descriptions from examples, in Winston, P.H., (editor), *The Psychology of Computer Vision*, McGraw-Hill, New York.
- [581] Winston, P.H., 1980. Learning and reasoning by analogy, *Communications of the ACM*, 23(12), 689-702.
- [582] Winston, P.H., 1982. Learning new principles from precedents and exercises, *Artificial intelligence*, 19(3), 321-350.
- [583] Winston, P.H., T.O. Binford, B. Katz and M. Lowry, 1983. Learning physical descriptions from functional definitions, examples, and precedents, in *Proceedings of AAAI'83*, Washington, D.C.
- [584] Winston, P.H., 1986. Learning by augmenting rules and accumulating censors, in R.S. Michalski, J.G. Carbonell and T.M. Mitchell (editors), *Machine Learning: An Artificial Intelligence Approach*, Morgan Kaufmann.
- [585] Wisniewski, E.J. and Anderson J.A., 1988. Some interesting properties of a connectionist inductive learning system, in *Proceedings of ICML-88*, San Mateo, CA.
- [586] Wollowski, M., 1989. A schema for an integrated learning system, in *Proceedings of IWML-89*, Cornell University, Ithaca, New York.
- [587] Wooldridge, M., N. R. Jennings. *Intelligent Agents: Theory and Practice*. *Knowledge Engineering*, 1995, 10(2)
- [588] Wu, Y., 1988. Reduction: A practical mechanism of searching for regularity in data, in *Proceedings of ICML-88*, San Mateo, CA.
- [589] Yao, Xin, Yong Xu, 2006. Recent Advances in Evolutionary Computation. *Journal of Computer Science and Technology*, 21(1): 1-18.
- [590] Yao, Y. Y., 2001. *Information granulation and rough set approximation*. *International Journal of Intelligent Systems* 16: 87-104

- [591] Ye, Shiwei, Zhongzhi Shi, Homotopy Scheme and Learning Vector Quantization, PRICAI-94, 1994.
- [592] Ye, Shiwei, Zhongzhi Shi, KL Transform and Learning Vector Quantization, ICNIP-94, Seoul, 1994.
- [593] Ye, Shiwei, Zhongzhi Shi, A Necessary Condition about the Optimum Partition on the Set with Finite Samples and Adaptive Frequency K-Means Clustering, PACES-95, pp. 1027-1031, 1995.
- [594] Ye, Shiwei, Zhongzhi Shi, Generalized K-means Clustering Algorithm and Frequency Sensitive Competitive Learning, Journal of Computer Science and Technology, Vol. 10, No.6, 1995.
- [595] Ye, Shiwei, Zhongzhi Shi, Neural Vector Quantization with Direct Sum Codebooks, ICYCS-95, 1995.
- [596] Yoo, J.P. and D.H. Fisher, 1989, Conceptual clustering of explanations, in Proceedings of IWML-89, Cornell University, Ithaca, New York.
- [597] Zadeh, L.A., *Fuzzy sets and information granularity*, in Gupta, M.M., Ragade, R.K. and Yager, R.R. (eds.), *Advances in Fuzzy Set Theory and Applications*, Amsterdam: North-Holland 1979
- [598] Zadeh, A. Towards a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. *Fuzzy Sets and Systems*. 19 (1997) 111-127
- [599] Zhang, J. and R.S. Michalski, 1989, A description of preference criterion in constructive learning: A discussion of basic issues, in Proceedings of IWML-89, Cornell University, Ithaca, New York.
- [600] Zhang J. 1992. Selecting Typical Instances in Instance-Based Learning. In Proceedings of the 9th International Machine Learning Workshop, Aberdeen, Scotland. Morgan Kaufmann, San Mateo, CA. 470 - 479
- [601] Zhang, Jian, Zhongzhi Shi, An Adaptive Theoretical Foundation Toward Unifying Neural Information Processing-----NFT, ICONIP-95, 1995.
- [602] Zheng, Z., Hu, H., Shi, Z.Z. Tolerance Relation Based Information Granular Space, Lecture Notes in Computer Science, Vol. 3641, pp. 682-691, 2005
- [603] Zytkow, J.M. and H.A. Simon, 1986, A theory of historical discovery: The construction of componential models, *Machine Learning*, 1(1), 107.
- [604] Zytkow, J.M. and A. Jankowski, 1988, Hierarchical control and heuristics in multisearch systems, *Methodologies for Intelligent Systems*, 4, 86-93.
- [605] Zytkow, J.M. and J. Baker, 1991, Interactive mining of regularities in databases, in G. Piatetsky-Shapiro and W. Frawley (editors), *Knowledge Discovery in Databases*, AAAI Press.
- [606] Zytkow, J.M. and Jieming Zhu, Automated empirical discovery in a numerical space, in Proceedings of CMLW'91, 1991.
- [607] 曹加恒, 1992, 机器学习理论方法与应用, 香港中华科技(国际)出版社.
- [608] 陈霖, 1986, 拓扑性质检测, 钱学森主编: 关于思维科学, 上海人民出版社.
- [609] 陈源, 1997, 约束推理与约束程序设计语言的研究, 硕士学位论文, 中国科学院计算技术研究所.
- [610] 郝付国, 1997, 模型和 CBR 相结合在中厚板生产中的应用, 硕士学位论文, 中国科学技术大学研究生院.
- [611] 李建会, 张江, 2006. 数字创世纪——人工生命的新科学. 科学出版社.
- [612] 李未, 1993, 一个开放的逻辑系统, 戴汝为、史忠植主编《人工智能和智能计算机》,

电子工业出版社, 1—6.

- [613] 李宝东, 1991. 基于范例的学习----- 模型和方法的研究, 硕士学位论文, 中国科学院计算技术研究所.
- [614] 李晓黎, 史忠植. 2000. 用数据采掘方法获取汉语词性标注规则. 计算机研究与发展, 37(12), 1409-1414
- [615] 廖乐健 (1994), 约束满足问题的研究, 博士学位论文, 中国科学院计算技术研究所.
- [616] 陆建江, 刘海峰. 2000. 数据库中广义模糊关联规则的挖掘. 工程数学学报. 17(1): 117-120
- [617] 陆汝铃, 1996. 人工智能. 科学出版社.
- [618] 吕翠英, 1994, 基于解释学习的石油蒸馏过程故障诊断机器学习系统, 博士学位论文, 石油大学研究生院.
- [619] 马海波, 1990, 解释学习的研究, 硕士学位论文, 中国科学院计算技术研究所.
- [620] 马洪文, 王万学, 李振江. 2000. 广义模糊关联规则的挖掘. 黑龙江商学院学报. 16(2): 4-9
- [621] 莫纯欢, 1996, 进化学习及人工生命中的突发性行为, 博士学位论文, 中国科学院计算技术研究所.
- [622] 钱学森, 1986, 开展思维科学的研究, 钱学森主编: 关于思维科学, 上海人民出版社.
- [623] 沈政、林庶芝, 1992, 脑模拟与神经计算机, 北京大学出版社.
- [624] 石纯一、黄昌宁等, 1993, 人工智能原理, 清华大学出版社.
- [625] 史忠植, 1988, 知识工程, 清华大学出版社.
- [626] 史忠植, 1990, 类思维的层次模型, 中国人工智能联合会议, 特邀报告.
- [627] 史忠植, 1990, 逻辑--对象知识模型, 计算机学报, No. 10.
- [628] 史忠植, 1990, 人类思维的层次模型, 第一届中国人工智能联合学术会议.
- [629] 史忠植、余志华, 1990. 认知科学和计算机. 科学普及出版社.
- [630] 史忠植, 1993a, 智能系统环境 INTSE, 中国科学院计算技术研究所.
- [631] 史忠植, 1993b, 神经计算, 电子工业出版社.
- [632] 史忠植. 智能主体及其应用. 科学出版社, 2000.
- [633] 史忠植. 知识发现. 清华大学出版社, 2001.
- [634] 史忠植. 高级计算机网络. 电子工业出版社. 2001.
- [635] 史忠植. 智能科学. 清华大学出版社, 2006.
- [636] 史忠植、张庆杰、张治洪、王军, 1995, 科学数据库知识发现, 全国科学数据库会议.
- [637] 史忠植、莫纯欢, 1995, 人工生命, 计算机研究与发展, Vol.32, No. 12.
- [638] 史忠植、陈源、廖乐健, 1996, 约束满足系统. '96 人工智能进展, 69-73.
- [639] 司马贺, 1986, 人类的认知, 科学出版社.
- [640] 宋志伟, 陈小平, 2003. 仿真机器人足球中的强化学习. 《机器人》, 24(7S):761-766.
- [641] 田启家、史忠植、王怀清, 1996, 多主体系统中的动作和知识推理, '96 人工智能进展, 74-79.
- [642] 田启家、史忠植, 1997, 动作和进化的逻辑基础, 中国科学.
- [643] 王军, 1997, 数据库知识发现的研究, 博士学位论文, 中国科学院计算技术研究所.
- [644] 王宽敬, 1993, 石油生产近期优化配产智能决策支持系统的研究与开发, 博士学位论文, 石油大学研究生院.
- [645] 汪涛、史忠植、田启家、王文杰, 1996. 现实世界中的主体的一种复合式结构. 软件学报.
- [646] 汪涛, 1996, 多主体处理环境 MAPE 中的主体结构、主体通信语言和主体构造方法,

硕士学位论文，中国科学院计算技术研究所.

- [647] 王文杰, 1998. 面向主体的软件开发环境 AOSDE. 博士学位论文, 中国科学院计算技术研究所
- [648] 王文杰, 叶世伟, 2004. 人工智能原理与应用. 人民邮电出版社.
- [649] 王学军, 1996, 多 Agent 系统中协调方法研究, 博士论文, 清华大学计算机系.
- [650] 王学重, 1990, 面向对象的炼油工程专家系统, 博士学位论文, 石油大学研究生院.
- [651] 徐众会, 1994, 基于范例推理的研究和在天气预报中的应用, 硕士学位论文, 中国科学院计算技术研究所.
- [652] 叶施仁. 2001. 海量数据约简与分类研究. 博士论文, 中国科学院计算技术研究所.
- [653] 叶世伟, 1995, 前向神经网络变换研究, 博士学位论文, 中国科学院计算技术研究所.
- [654] 张海俊, 2005. 基于主体的自主计算研究. 博士学位论文, 中国科学院计算技术研究所
- [655] 张建、史忠植, 1995, 迈向计算智能的统一理论框架, 中国科学协第二届青年学术会议.
- [656] 张建, 1996, 基于微分流形神经场计算理论及其在金融分析中的应用, 博士学位论文, 中国科学院计算技术研究所.
- [657] 张钺, 张铃, 1990. 问题求解的理论与应用. 北京: 清华大学出版社.
- [658] 张铃, 张钺. 模糊商空间理论(模糊粒度计算方法). 软件学报, 14(4)2003: 770-776.
- [659] 张治洪、史忠植、张庆杰、谭宁, 1996, WWW 与数据库的连接, 计算机研究与发展.
- [660] 周涵, 1993, 基于泛例学习的内燃机油产品设计系统, 博士学位论文, 石油大学研究生院.
- [661] 左万利, 刘居红. 1999. 任意多表间关联规则的并行挖掘. 吉林大学自然科学学报. Vol.4
- [662] 朱朝晖、戈也挺、陈世福等, 关于行动推理的研究. 陆汝钤主编, 世纪之交的知识工程与知识科学, 清华大学出版社, 2001